PhD-FSTC-2015-26
The Faculty of Sciences, Technology and Communication

DISSERTATION

Defense held on 12/05/2015 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

by

Srinivas Vivek VENKATESH
Born on 7 September 1985 in Mysore (India)

PRACTICAL PROVABLE SECURITY AGAINST SIDE-CHANNEL ATTACKS

Dissertation defense committee

Dr. Jean-Sébastien Coron, dissertation supervisor
Associate Professor, Université du Luxembourg

Dr. Pierre-Alain Fouque
Professor, Université de Rennes 1, France

Dr. David Galindo
Scytl, Barcelona, Spain

Dr. Volker Müller, Chairman
Associate Professor, Université du Luxembourg

Dr. Martijn Stam, Vice Chairman
University of Bristol, United Kingdom

Abstract

Securing cryptographic systems in the presence of side-channel leakages is still an important problem. Over recent years, the cryptography theory community has shown considerable interest in formally modelling the side-channel leakages and in designing "provably" secure cryptographic primitives in these leakage models. This area is often referred to as *leakage-resilient cryptography*. Yet, designing a formal model that realistically captures side-channel leakages such as power consumption patterns, and designing primitives efficient enough to be deployed in practice in such a leakage model, remains a challenging research direction.

In this work, we aim to bridge the above gap between the theory of provably secure cryptosystems that resist side-channel attacks and their practical relevance. Keeping this goal in mind, we analyze existing constructions and provide new ones for basic cryptographic primitives such as encryption and authentication in both public-key and symmetric-key cryptography.

This dissertation consists of three parts. In the first part, we analyze existing and design new efficient leakage-resilient constructions for public-key encryption and digital signatures that tolerate *continual* leakage in the *split-state* leakage model. Our security reductions are in the generic bilinear group model. The constructions we consider are simple variants of the ElGamal key encapsulation mechanism, and the Boneh–Boyen and the Schnorr signature schemes. We also cryptanalyze a variant of the ElGamal key encapsulation mechanism that was previously conjectured to be leakage-resilient under certain conditions.

The second part of this work is concerned with the protection of block ciphers in the *probing* adversarial leakage model. This approach, popularly known as *masking* in the cryptographic engineering community, is an effective countermeasure for block cipher implementations against power-analysis attacks. We improve the efficiency of a generic higher-order masking scheme recently proposed by Carlet et al. Improving the efficiency of this scheme is related to the problem of evaluating polynomials over binary finite fields in a newer cost model that counts only "non-linear" polynomial multiplications. We propose a new method for evaluating polynomials in this cost model, and argue (heuristically) that this method is asymptotically optimal.

The third part deals with the construction of efficient leakage-resilient symmetric-key authentication and encryption schemes. The constructions are shown to be secure in the standard model under a recently introduced *simulatable leakage assumption*. This assumption offers practitioners a hope to work with a formal leakage model that allows empirical verifiability. We propose a leakage-resilient CBC-like message authentication code, and also propose a leakage-resilient PRG-based chosen-plaintext secure encryption scheme for which we quantify the leakage it tolerates during the *challenge phase* in terms of security of a single iteration. Our constructions tolerate continual leakage but require leak-free updates.

Dedicated to

Bhagavān Śrī Kṛṣṇa

Acknowledgements

First and foremost, I am grateful to Jean-Sébastien Coron and David Galindo for providing me a great opportunity to pursue this doctoral dissertation under their supervision. Jean-Sébastien has been a great source of inspiration for excellent research and has provided me ample freedom to pursue research problems of my interest. David has guided me in this journey at every stage and this dissertation would not be possible but for his supervision. It has been a great learning experience working with both of them.

I would like to thank Alex Biryukov for serving on my dissertation supervisory committee. I would also like to thank Pierre-Alain Fouque, Volker Müller and Martijn Stam for serving on my dissertation defense committee.

I would like to thank all my other co-authors Johann Großschädl, Zhe Liu, Olivier Pereira, Arnab Roy, François-Xavier Standaert, and Praveen K. Vadnala for great collaborations. Special thanks to Olivier Pereira and François-Xavier Standaert for providing me a great opportunity to visit the Université catholique de Louvain (UCL) Crypto group during Summer 2014.

The Laboratory of Algorithmics, Cryptology and Security (LACS) in the University of Luxembourg has always been providing an excellent research and joyful working environment. I would like to thank all my present and former colleagues at LACS, and other friends in Luxembourg for a wonderful time. Special thanks to Fabienne Schmitz for all her help in the administrative tasks. I would also like to thank all the members of the UCL Crypto group for a very nice time during my visit.

Finally, my deepest gratitude to my parents, brother, and wife, for all their love, care and support.

Srinivas Vivek Venkatesh, May 2015

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Contents

1.1 Information Security and Cryptography

Securing information against unintended access and use has been a concern since the beginning of human civilization. With the advent of modern telecommunication systems and digital computers, the ability to generate, process, transmit and store information has tremendously improved. Needless to say, the need to protect the information in digital form has also become increasingly vital. Some of the *information security* objectives are as follows: *privacy or confidentiality, data integrity, entity authentication or identification, message authentication, signature, authorization, validation, access control, certification, time-stamping, witnessing, receipt, confirmation, ownership, anonymity, non-repudiation,* and *revocation* [MvV96, Section 1.2].

Of the above objectives, the following four are the main ones through which others can be realized:

- *confidentiality*: to protect data from unauthorized access,
- *data integrity*: to protect data from unauthorized alteration,
- *authentication*: to correctly identify communicating entities and/or identify the origin of data,
- *non-repudiation*: to protect against denial of previous commitments.

Cryptography is a discipline that provides the mathematical foundations and techniques that are needed to realize the above goals of information security. Cryptography has an interesting history and it has been known to be used since the Roman times. It has also played a crucial role in the outcome of the World Wars. We refer to [Kah96] for a comprehensive account of the history of cryptography.

Cryptography can be broadly classified into *symmetric-key cryptography* and *public-key cryptography*. In the former the entities involved share a common secret key, while in the latter no common secret is needed. While symmetric-key cryptography has been in use since historic times, the commonly accepted origin of public-key cryptography can be traced to the seminal work of Diffie-Hellman [DH76].

Some of the most basic symmetric-key primitives are symmetric-key encryption ciphers - *stream ciphers* and *block ciphers*, and *message authentication codes* (MACs). In public-key cryptography the corresponding primitives are *public-key encryption schemes* and *digital signatures*. A *hash function* is an important unkeyed cryptographic primitive.

1.2 Cryptanalysis

The study of techniques to defeat the goals of cryptographic primitives and protocols is termed as *cryptanalysis*. Typically, the goal and the capabilities of an adversary attacking a cryptosystem will be clearly defined. Adversaries can be broadly classified into two types: *passive* and *active*. In the former scenario, an adversary merely observes the communications and/or the computations, while in the latter they try to influence the execution of the cryptosystem.

Traditional attack scenarios (i.e., adversarial capabilities) for encryption schemes in both the symmetric-key and public-key settings are: *ciphertext-only attack, known-plaintext attack, chosen-plaintext attack, adaptive chosen-plaintext attack, chosen-ciphertext attack,* and *adaptive chosen-ciphertext attack* [MvV96, Section 1.13]. The most common adversarial goal is to distinguish the encryptions of any two arbitrary messages of the same length [GM84].

In the case of MACs and digital signature schemes, the usual attack scenarios are: *key-only attack, known-message attack, chosen-message attack,* and *adaptive chosen-message attack*. The usual adversarial goals are: *selective forgery* and *existential forgery* [Sti95, Section 7.2].

We shall describe some of the above attacks in detail at appropriate places in later chapters.

1.2.1 Implementation Attacks: Side-Channel Attacks

The above traditional attack scenarios on basic crypto primitives do not allow an adversary to obtain information from the secret internal state of an execution. However, it is possible to successfully mount attacks on cryptographic implementations that recover (possibly partial) information about the internal state [Koc96, KJJ99]. Such attacks are particularly feasible on embedded systems such as smart cards, and hence protecting these devices against such attacks is often necessary. Some of the most commonly exploited sources of these "side-channel leakage" are *timing information* [Koc96], *power consumption* [KJJ99], and *electromagnetic emission* [QS01]. Less commonly used leakage sources are *acoustic emanation* [GST13], *optical emission* [FH08], and *thermal emissions* [BKMN09].

Implementation attacks, such as those mentioned above, that are based on observation of physical leakages, are termed *side-channel attacks*. Other than side-channel attacks, another important class of implementation attacks are *fault attacks* [BDL01, BS97]. Fault attacks typically exploit the erroneous outputs induced by a fault in a computation to recover some secret information.

Implementation attacks, not necessarily on cryptosystems, have an interesting history dating back to the World Wars. However, it is not until the works of [Koc96, KJJ99] that these types of attacks got wide spread attention in the cryptographic community. For a brief history of implementation attacks, we refer to [Kiz11, Section 1.3].

Broadly speaking, implementation attacks are categorized into two types: *passive* and *active*, analogous to the black-box setting. In passive implementation attacks, an adversary merely observes an execution of an implementation that nearly works as intended, whereas, in active attacks, an adversary tries to influence the execution of an implementation causing it to deviate from its normal execution. Side-channel attacks belong to the category of passive attacks, while fault attacks are active attacks.

Alternatively, implementation attacks can be also classified as: non-invasive, semi-invasive, and invasive, depending upon to what extent an implementation is intruded. Side-channel attacks are mostly non-invasive, but sometimes are required to be semi-invasive, for instance, by partially removing the packaging of an integrated circuit to facilitate the observation of electromagnetic emissions.

After timing attacks, the next most feasible side-channel attacks on cryptographic implementations are power analysis attacks. Power analysis attacks can broadly classified into: *simple power analysis* attacks, *differential power*

analysis attacks, and *template* attacks. In the following, we briefly describe these three attacks. For a more detailed account of side-channel analysis, in particular, the power analysis attacks, we refer to [MOP07].

Typical stages in a power analysis attack, and generally for any side-channel analysis experiment, are: *profiling stage*, *online stage*, and *offline stage* [Kiz11, Section 1.3]. In the profiling stage, which is optional, an adversary sets up a characteristic profile for the leakage of a device which it has access to. In the online stage, the adversary performs the measurements to obtain the side-channel race. In the final stage, the offline stage, all the data gathered is analyzed to (usually) recover the secret key.

Leakage from a power analysis experiment, and also generally, will firstly be dependent on the device in hand. Additionally, it will be operation dependent and/or data dependent. Every measurement data will be accompanied with noise. In many attacks, to associate the leakage to intermediate variables, we need a "leakage model". In practical analysis, a device is typically assumed to leak the *hamming weight* of intermediate variables or leak the *hamming distance* of the old and new values of a variable. Typically the goal of power analysis attacks, and also generally, is to recover the secret key.

1.2.1.1 Simple Power Analysis

In a Simple Power Analysis (SPA) attack an adversary tries to deduce a secret key by analyzing a single trace, or many traces individually. A well-known SPA attack scenario is on modular exponentiation implemented in a straightforward way using square-and-multiply technique, which is widely used in public-key cryptography [KJJ99]. SPA attacks on symmetric-key cryptographic implementations are also known to be effective [Man02].

1.2.1.2 Differential Power Analysis

Differential Power Analysis (DPA) attacks are based on first obtaining several power traces corresponding to many executions using the same fixed secret key [KJJ99]. It is a divide-and-conquer-based approach that recovers the secret key one part at a time. DPA attacks are effective against symmetric-key implementations, which are typically deterministic. Various statistical tools are used to analyze the resulting leakage trace and recover the corresponding chunks of the secret key [MOP07, Chapter 6]. DPA attacks exploit a single point (i.e., sample) in a leakage trace corresponding to a targeted variable. In higher-order DPA attacks, multiple points in a leakage trace corresponding to a targeted variable are exploited, and hence protection against such attacks can be more challenging [Mes00]. Nevertheless, mounting even third-order attacks is already challenging.

1.2.1.3 Template Attacks

Template attack is an another divide-and-conquer-based strategy to mount a power analysis attack when adversary has the option do profiling [CRR02]. Here, an adversary builds a collection of templates corresponding to known values of a targeted intermediate variable during the initial profiling stage. Once leakage traces are obtained, they are then compared against the template collection using statistical tests. Template attacks can be considered to be one of the most powerful form of power analysis attacks, and they do not necessarily need a model for leakage. But they require that adversary has full access to a copy of targeted device.

1.3 Provable Security

In the previous section, we saw various attack scenarios against cryptosystems ranging from black box cryptanalysis to implementation attacks. Designing cryptoprimitives that are secure against a given set of attacks and formally analyzing their security is an important task. Such rigorous study of cryptosystems dates back to the work of Shannon in the 1940s [Sha49]. He introduced the notion of *perfect secrecy* and mathematically argued that the one-time pad cipher satisfies such a property, that is, the one-time pad resists ciphertext-only attack. However, to achieve such a property, the key size has to be at least as long as the size of messages, which is impractical in modern communications.

The seminal work of Goldwasser and Micali [GM84] introduced the notion of *semantic security* for encryption schemes which may be viewed as an analogue, in the computational setting, of the notion of (information-theoretic) perfect secrecy. Semantic security requires that any adversary is not able to learn any efficiently computable function of the underlying message given only the ciphertext. This notion was shown to be equivalent to the notion of *indistinguishability*. These notions and their generalization to other primitives allowed practical construction of cryptoprimitives that allowed "security proofs" under certain computational assumptions.

The paradigm of *provable security* aims at the formal design and analysis of cryptosystems by first formally specifying the adversarial attack model. Then, the security of a designed scheme is argued via a *reduction* from an usually well-accepted computational assumption or from the assumed security of a smaller primitive. Namely, one argues that if there exists an adversary against the designed cryptosystem, then one could construct an algorithm (resp., adversary) against the computational assumption (resp., smaller primitive). The security definitions typically will be either *attack-game based*, such as the indistinguishability game for encryption schemes, or *simulation based*, such as the security definitions used in multi-party computation. For a detailed in-

troduction to the provable security approach to cryptography, we refer to the books [Gol01, Gol04, KL07].

A computationally efficient adversary is specified either as a *probabilistic polynomial-time turing machine* or a *family of polynomial-sized boolean circuits*. Widely used computational assumptions are the existence of *one-way functions*, *psuedorandom generators* and *psuedorandom functions*. In public-key cryptography, intractability of *integer factorization problem*, *discrete logarithm problem* and certain *lattice-based problems* are widely used.

Often security reductions need to make use of certain idealized model of computations in order to obtain security proofs in some instances. Two such popular idealized models are the *random oracle model* [FS86] and the *generic group model* [Sho97]. In the random oracle model, a hash function is assumed to behave as a random function. Whereas in the generic group model, the group elements are assumed to have random encodings. Since we use the generic group model extensively in Part I of this dissertation, we will briefly elaborate on this model.

Generic Group Model. The generic group model was introduced in the seminal work of Shoup [Sho97]. As mentioned before, group elements are generally assumed to have random encodings, though some variants of this model do not need this random encodings requirement, for instance, see [Mau05]. Intuitively, in this model an adversary cannot exploit the representations of group elements other than checking for equality of the elements. A security proof in the generic group model only rules out "generic attacks," that is, it rules those attacks that work over any instantiation of the underlying group.

Generalizations of the generic group model are also widely used. One such extension that we use in our work is for pairing-based groups, called the *generic bilinear group model* [BB04]. We defer a formal description of this model to Section 2.2.2.3.

The generic group model is often criticized (sometimes rather unfairly) as being too idealistic to be practically relevant. For a discussion on this issue, we refer to [KM07].

1.3.1 Leakage-Resilient Cryptography

In Section 1.2.1, we saw that side-channel attacks are a security concern for implementations on embedded devices. Various types of countermeasures both at the hardware/physical and software/algorithmic level have been proposed. For different classification of these countermeasures, we refer to [MOP07], [Kiz11, Section 1.3], and [Fau10, Section 3.3].

In this dissertation, we consider a class of algorithmic countermeasures that are usually referred to as being *leakage resilient*. Such countermeasures address the issue of side-channel leakage at the design stage of a cryptographic

primitive itself. For doing so, we need to specify a *leakage model* that "reasonably" incorporates the side-channel leakages that we wish to protect against.

In Section 1.2.1, we saw that timing attacks and power analysis attacks are a serious security threat for implementations on embedded devices. Timing attacks are simpler to prevent than power analysis attacks. Here we usually need to ensure that all the execution paths are of uniform length, say, by using dummy instructions. In contrast, power analysis attacks can leak intermediate variables and ensuring protection against such attacks is a serious concern.

Next, we describe some previously used leakage models that we consider in this work. Though our primary concern is protection against power analysis attacks, we believe that in most cases they reasonably model a broader class of side-channel leakages, that arguably includes electromagnetic emanations.

1.3.1.1 Split-State Leakage Model

This model is based on the *only computation leaks* paradigm introduced in the work of Micali and Reyzin [MR04]. In this model computations are divided into steps that leak independently. The split-state model is a slightly powerful leakage model that only requires that the internal memory is usually split into two parts that leak independently, and not necessarily only computation leaks [LL12]. The leakage in these models is specified as an efficiently computable function of the secret state, i.e., the secret key and the internal randomness. The output length of these leakage functions is bounded. An adversary can obtain leakages either a bounded number of times, or overall it can be unbounded, i.e., *continual leakage*. Typical justification of the bounded length requirement of leakage functions is that in practice many side-channel attacks only exploit a few bits of leakage information for reasonable implementations [KP10a].

Some variants of the split-state model do not put bounded length requirement for the leakage functions. Instead, they require that the remaining *min-entropy* or the *computational entropy* is sufficiently high. There are already several leakage-resilient schemes that are proposed in the split-state model and its variants, some of which are [DP08, Pie09, KP10a, FKPR10, LL12].

One important remark is that most of the public-key primitives that are leakage-resilient in this leakage model and have proofs of security in the "standard model" are inefficient to be implemented on embedded devices.

1.3.1.2 Probing Leakage Model

This leakage model was introduced by Ishai, Sahai and Wagner [ISW03]. In this model, originally introduced for boolean circuits, any adversary could probe a bounded number of wires, say t, and read the values carried. A general

circuit compiler was proposed that transforms a given circuit into a leakage-resilient circuit. The size of the resulting circuit is increased by a factor of t^2. The secret-sharing techniques are employed to obtain the transformed circuit. Later this model was extended to computation over finite fields [RP10].

The technique of secret sharing (a.k.a. *masking*) has long been employed in cryptographic engineering to protect implementations against side-channel attacks [CJRR99]. Provably secure masking schemes of arbitrary order exist that are based on this leakage model [CGP$^+$12, Cor14][1].

Recent works extend the probing model to the case where all the intermediate variables leak but the leakage is noisy [PR13, DDF14]. Such a leakage model captures practical scenarios more realistically.

1.3.1.3 Simulatable Leakage Assumption

This leakage model was recently introduced by Standaert, Pereira and Yu [SPY13] in the context of symmetric-key primitives that use block ciphers. This framework offers practitioners a hope of placing reasonable restrictions on leakage functions that are empirically verifiable in side-channel evaluation laboratories, unlike most of the known leakage models. A leakage-resilient psuedorandom generator was constructed in [SPY13], with a relatively simple security proof in the standard model.

Other Leakage Models. We briefly mention other leakage models that were previously considered, mainly by the cryptography theory community. For a survey of these models, we refer to [Wic11, Chapter 2].

One such widely considered model is the *full memory leakage model* that allows leakage functions to take as input the entire secret state [AGV09, NS09]. Variants of this scheme, similar to the split-state model, differ in terms of types of leakages allowed. Other models that allow such full leakage are found in works related to *threshold cryptography* [DF89], *key-insulated cryptography* [DKXY02], and *forward-secrecy* [DvOW92]. Some other known partial leakage models are *oblivious RAMs* [Gol87] and models in *exposure-resilient cryptography* [Dod00].

As remarked earlier for the split-state model, most of the public-key primitives whose security proofs do not resort to an idealized computation model, such as the random oracle or the generic group models, are inefficient to be implemented in practice.

[1] Sometimes provably secure masking schemes, such as those based on the probing leakage model [ISW03], are not classified as leakage-resilient countermeasures. But we feel it is not inappropriate to consider them as being leakage resilient.

1.4 Our Contribution and Organization

In this work, we aim to bridge the gap between the theory and practice of leakage-resilient cryptography, in both the public-key and symmetric-key settings.

This dissertation consists of three parts.

- In Part I, we analyze existing and propose new efficient leakage-resilient constructions for public-key encryption and digital signatures that tolerate continual leakage in the split-state leakage model, where even key updates are allowed to leak. All the above schemes are simple variants of existing well-known schemes obtained by *blinding* (i.e., splitting) the secret key. Our security proofs are in the bilinear generic group model. Arguing about their security in the standard model is currently out of reach of existing techniques [Wic13].
  - Chapter 2 contains two main results. First, we cryptanalyze a variant of the ElGamal key encapsulation mechanism by Kiltz and Pieterzak [KP10a] that was conjectured to tolerate continual leakage in the split-state model. We give a non-trivial upper bound on the amount of leakage tolerated by this conjecture, by exhibiting an attack that recovers the full secret key. The attack uses a new variant of the *hidden number problem*, that we call *hidden shares - hidden number problem*. This part is based on the publication [GV14].
    As a second contribution, we provide an improved security analysis of a bilinear variant of the ElGamal key encapsulation mechanism, also due to [KP10a]. We give an improved security analysis of this scheme. Next, for practical considerations, we modify this scheme. Then we implement our variant and analyze its side-channel resistance. This part is based on the publication [GGL+14].
  - In Chapter 3, we propose a leakage-resilient signature scheme in the continual split-state leakage model that is based on the identity-based encryption scheme by Boneh and Boyen [BB04]. Our scheme is existentially unforgeable against *adaptive* chosen message and leakage attacks. This chapter is based on the publication [GV12].
  - In Chapter 4, we propose a leakage-resilient pairing-based variant of the Schnorr signature scheme. This scheme also enjoys properties similar to the scheme from Chapter 3. This chapter is based on the publication [GV13].
- In Part II (Chapter 5), we improve the efficiency of a generic higher-order masking scheme proposed by Carlet et al. [CGP+12]. This scheme is secure in the probing leakage model [ISW03]. Efficiency of this scheme is related to the problem of evaluating polynomials over binary finite fields representing S-boxes. The corresponding evaluation cost model considers only *non-linear multiplications*.

We take a formal approach to this polynomial evaluation problem. We investigate optimal methods for exponentiation in $\mathbb{F}_{2^n}$ by studying a previously proposed variant of addition chain that we call as *cyclotomic-class addition chain*. We define the notion of $\mathbb{F}_{2^n}$-polynomial chain for arbitrary polynomials, and use it to count the number of non-linear multiplications required to evaluate polynomials over $\mathbb{F}_{2^n}$. Among several interesting properties, we prove lower bounds for optimal evaluation methods. We propose a new heuristic technique for evaluating arbitrary polynomials, whose complexity is asymptotically optimal. In practice, with this new technique, we significantly reduce the complexity of evaluating S-boxes compared to previous methods.

This chapter is based on the publications [RV13] and [CRV14]. A part of the results in [RV13] has also appeared in the dissertation of Arnab Roy [Roy13].

– In Part III (Chapter 6), we construct efficient leakage-resilient symmetric-key authentication and encryption schemes in the standard model under the *simulatable leakage assumption* [SPY13].

  We propose a leakage-resilient CBC-like message authentication code, and also propose a leakage-resilient PRG-based chosen-plaintext secure encryption scheme for which we quantify the leakage it tolerates during the *challenge phase* in terms of security of a single iteration. Our constructions tolerate continual leakage but requires leak-free updates. This chapter is based on the manuscript [PSV15].

1.5 Notation

In this dissertation, we denote the set of integers by $\mathbb{Z}$, and $\mathbb{N}$ denotes the set of positive integers. By $\mathbb{Z}_p$ (resp. $\mathbb{Z}_q$) we denote, depending upon the context, either the set of integers $\{0, 1, \ldots, p-1\}$ or the ring modulo p (resp. q), where $p, q > 0$. Unless otherwise specified, p and q are prime numbers. The field of p^n elements is denoted by $\mathbb{F}_{p^n}$. The field $\mathbb{F}_p$ is identified with $\mathbb{Z}_p$, unless evident from the context.

We denote a random sampling of an element $a \in A$ from a set A, and also denote a (possibly probabilistic) output of an algorithm A, by $a \leftarrow A$. If we want to explicitly denote the randomness r used during the sampling/output, then we do so by $s \xleftarrow{r} S$. Unless otherwise mentioned or implicit from the context, any sampling is from an uniform independent distribution. This is always implied by the notation $s \xleftarrow{\$} S$.

The symbol ":=" is used to define a notation in an expression, as in $A := \mathbb{Z}$, or to explicitly indicate an output of a deterministic algorithm or a function.

Chapter specific notations are defined in respective chapters.

Part I

Split-State Leakage Model: Cryptanalysis & Efficient Public-Key Constructions

Chapter 2

Split-State ElGamal Encryption

In this chapter, we first describe an attack on a variant of the ElGamal key encapsulation mechanism (EG-KEM) proposed in [KP10a]. It was conjectured in [KP10a] that the said EG-KEM with stateful decryption resists lunch-time chosen ciphertext (i.e., CCA1) and leakage attacks in the split-state model We give a non-trivial upper bound on the amount of leakage tolerated by this conjecture. More precisely, we show that the conjecture does not hold if more than a $\left(\frac{3}{8} + o(1)\right)$ fraction of the bits are leaked at every decryption step, by showing a lunch-time attack that recovers the full secret key. The attack uses a new variant of the *hidden number problem*, that we call *hidden shares - hidden number problem*, which is of independent interest.

Secondly, we provide an improved security analysis of a bilinear variant of the ElGamal KEM (BEG-KEM), also due to [KP10a]. More precisely, we weaken the restriction on the image size of the leakage functions in the previously used bounded leakage model, and only insist that the inputs to the leakage functions have sufficient min-entropy left, in spite of the leakage, with no limitation on the quantity of this leakage. Such a model is closer to practice than the bounded leakage model. We provide a novel security reduction for BEG-KEM in this relaxed leakage model using the generic bilinear group model. Finally, consider the problem of implementing this scheme securely in practice. We propose, implement and evaluate an advanced variant of BEG-KEM, BEG-KEM+, that generates random elements in the pairing base group, avoiding exponentiation by a secret integer value, but rather uses an encoding to the pairing base group due to [FT12]. The implementation is in software and the side-channel evaluation (w.r.t. power analysis attacks) is based on an exhaustive survey of the side-channel cryptanalysis literature.

Contents

2.1 Limits of a Conjecture

2.1.1 Introduction

Leakage-resilient cryptography is a recent research line that aims at building countermeasures and/or defences against side-channel attacks while providing security reductions in a provable manner [ISW03, MR04, DP08, AGV09,

NS09, DGK$^+$10, DHLAW10a, SPY13]. The methodology is as follows: an abstract model specifying the leakage data is chosen and the goal is to exhibit a reduction from a hardness assumption to a hypothetical side-channel adversary. The theory of leakage-resilient cryptography has witnessed a tremendous activity despite its short life.

However, meeting the strongest security levels, such as, resiliency against a broad class of continual leakage attacks [DHLAW10a], under the weakest assumptions, say, memory leakage model [AGV09] or auxiliary input model [DGK$^+$10], is still out of reach from a practical point of view. To our knowledge existing schemes are inefficient when compared to their counterparts in the non-leakage setting; moreover, current leakage-resilient constructions are conceptually far more complex than those a practitioner currently finds in its cryptographic tool-box. Indeed, in an important work, Wichs [Wic13] shows that it might be impossible to achieve leakage-resilience for cryptosystems whose secret key is uniquely determined by its public key, unless we weaken the security model. The former property is enjoyed by most practical cryptosystems nowadays.

In this sense it is worth to mention the work by Kiltz and Pietrzak [KP10a]. They propose BEG (also denoted BEG-KEM), a pairing-based analogue of the ElGamal Key Encapsulation Mechanism (EG-KEM) [Gam85] with stateful decryption that is leakage-resilient against lunch-time chosen ciphertext and leakage attacks (CCLA1). The latter means that the classical distinguishing adversary against ElGamal is given access to decryption and leakage oracles only before the challenge ciphertext is given. The basic idea is to set the ElGamal secret key to be a group element (in contrast to an integer), and then multiplicatively share it. While splitting the secret key before decryption is a well-known technique from the side-channel countermeasures literature, the novelty of this work is to propose to split a *group element* rather than an *integer*.

The work [KP10a] does not contradict the impossibility result by Wichs. Indeed, [KP10a] slightly weakens the security model by using the split-state model [MR04], meaning that the computation can be divided into steps, where each such step accesses different parts of the memory that leak independently. Their security arguments hold in an extension of the generic group model [Sho97] called generic bilinear group (GBG) Model [BBG05]. Still, they consider leakage to be *continual leakage*, i.e. the useful leakage data per decryption invocation is bounded in length, but unbounded overall. More concretely, it is shown that if the secret key length is κ, then their scheme is secure against leakage of at most $\lambda \ll \frac{\kappa}{2}$ bits at every decryption step. More precisely, $\lambda < \frac{\kappa}{2} - \omega(\log \kappa)$ to make it infeasible to guess the remaining bits of the secret key by a brute force attack.

The authors of [KP10a] discuss the limitations of getting a security proof for a similar leakage-resilient property for EG-KEM with stateful decryption over *arbitrary groups* where the decisional Diffie-Hellman problem is believed to

be hard. Nevertheless they conjecture that, for certain such arbitrary groups, EG-KEM with stateful decryption, denoted $\mathsf{EG}^\star$, might be resistant against side-channel attacks that abide by the continual leakage split-state model.

2.1.1.1 Our Contribution

In this work we impose a limit on the conjecture from [KP10a], by proving that if a minimum of $\left(\frac{3}{8} + o(1)\right)\kappa$ bits are leaked at every invocation of the secret key, a CCLA1 attack exists against EG-KEM with stateful decryption scheme $\mathsf{EG}^\star$ that recovers the secret key. Naturally our limit on the conjecture is built using the abstract model of leakage specified in the conjecture, that is, the only computation leaks information model. Interestingly, our limit to the conjecture applies to any arbitrary instantiation of the underlying group. To achieve this result we define a new problem, called hidden shares - hidden number problem, which is a close but new variant of the hidden number problem [BV96].

2.1.1.2 Known Hidden Number Problems

The hidden number problem (**HNP**) was originally introduced by Boneh and Venkatesan [BV96] to demonstrate the hardness of computing the most significant bits of the secret key in the Diffie-Hellman key exchange mechanism. In its most generic form it can be described as follows. Let

$$f_\alpha : \mathcal{D} \to \mathcal{V}, \qquad \alpha \in \mathcal{A}$$

be a family of maps between algebraic domains $\mathcal{D}$ and $\mathcal{V}$. The map is parametrized by α, that takes values from some set $\mathcal{A}$.

Definition 2.1 [Generic HNP [Shp05]] Suppose some partial information about $f_\alpha(t) \in \mathcal{V}$ is given for several values of t (also completely specified), chosen uniform randomly from a subset $\mathcal{T} \subseteq \mathcal{D}$, find α.

Typically, $\mathcal{D}$ and $\mathcal{V}$ are finite fields $\mathbb{F}_p$. An instance of the Generic HNP problem is the Modular Inversion Hidden Number Problem (MIHNP) [BHHG01], also called $\mathbb{F}_p$-Inverse-HNP [Shp05]. In MIHNP, we have $\mathcal{D} = \mathcal{T} = \mathbb{F}_p \setminus \{-\alpha\}$, $\mathcal{V} = \mathcal{A} = \mathbb{F}_p$, where

$$f_\alpha(t) = \mathrm{MSB}_{k,p}\left(\frac{1}{\alpha + t}\right),$$

where $\mathrm{MSB}_{k,p}(z)$ means the (integer representing the) k most significant bits of $z \pmod p$, and the elements of $\mathbb{F}_p$ are identified with integers of fixed bit-length $\lfloor \log_2 p \rfloor + 1$. In other words, we are given $n+1$ pairs $(t_i, \mathrm{MSB}_{k,p}(1/(\alpha + t_i)))$, for $i = 0, \ldots, n$, where $t_i \in \mathbb{F}_p \setminus \{-\alpha\}$ are chosen uniform randomly and independently, and the goal is to find a polynomial-time (in $\log p$) algorithm to recover $\alpha \in \mathbb{F}_p$ completely. The hardness of variants of this problem has

been used to construct efficient algebraic PRNGs and MACs [BHHG01]. A close variant of MIHNP is one where

$$f_{\alpha,\beta}(t) = \mathrm{MSB}_{k,p}\left(\frac{\beta}{\alpha + t}\right).$$

Another problem related to the HNP was addressed in [HGNS03] and it is called the "HNP with hidden multipliers" (HM-HNP). In [Shp05], the same problem is referred to as $\mathbb{F}_p$-Approx-HNP.

Definition 2.2 [HM-HNP [HGNS03]] Given $n+1$ pairs $(\mathrm{MSB}_{k,p}(t_i), \mathrm{MSB}_{k,p}(\alpha \cdot t_i))$, where $i = 0, \ldots, n$, $\alpha \in \mathbb{F}_p$ and $t_i \xleftarrow{\$} \mathbb{F}_p$, find α.

In HM-HNP, we need to find $\alpha \in \mathbb{F}_p$ given $(\mathrm{MSB}_{k,p}(t_i), \mathrm{MSB}_{k,p}(\alpha t_i))$, where $t_i \in \mathbb{F}_p$. This variant is related to proving the bit security of "time-released crypto", among other applications. Variants of the HNP have also been used to attack DSA [HGS01, NS02].

2.1.1.3 Hidden Shares - Hidden Number Problem

We propose a variant of the MIHNP and HM-HNP problems that we call Hidden Shares - Hidden Number Problem (HS-HNP).

Definition 2.3 [HS-HNP] Given $n+1$ pairs $\left(\mathrm{MSB}_{k,p}(t_i), \mathrm{MSB}_{k,p}\left(\frac{\alpha}{t_i}\right)\right)$, where $i = 0, \ldots, n$, $\alpha \in \mathbb{F}_p$ and $t_i \xleftarrow{\$} \mathbb{F}_p \setminus \{0\}$, find α.

Note that unlike the Generic HNP, in HS-HNP (also HM-HNP) only a partial information about the values of (random) t_i is given. The name "hidden shares" follows from the fact that t_i and $\frac{\alpha}{t_i} \pmod p$ are two multiplicative shares of $\alpha \in \mathbb{F}_p$. To our knowledge the HS-HNP has not been explicitly addressed before. However, as is the case with most variants of the HNP, our approach uses lattice techniques [Cop97, BHHG01]. Particularly, we use the techniques of [BHHG01].

We stress that in spite of similarities in the techniques used to solve various variants of HNP, there are significant differences in the technical details. For example, the exact structure of the lattices set up in the solution vary according to the problem. These differences eventually reflect in the amount of partial information (value of k) required to solve the problem with a reasonable success probability. For comparison, the values of k currently required for HNP, MIHNP, and HM-HNP are $\sqrt{\log p} + \log\log p$, $\frac{1}{3}\log p$, and $\frac{4}{5}\log p$, respectively [BV96, BHHG01, HGNS03]. As we shall see in Section 2.1.3, HS-HNP currently requires the value of k to be $\frac{3}{4}\log p$.

The HS-HNP can be generalized to arbitrary commutative groups.

Definition 2.4 [Generalized HS-HNP] Let $(\mathbb{G}, \cdot)$ be a group and $\alpha \in \mathbb{G}$. Given partial information on each $t_i \in \mathbb{G}$ and $\alpha \cdot t_i^{-1}$, for $i = 0, \ldots, n$, find α.

If $(\mathbb{G}, \cdot)$ is modelled as a "generic group" [Sho97], then the Generalized HS-HNP is hard, provided that the shares have sufficient remaining entropy. This is a straightforward consequence of the work in [KP10a] and also of that in this dissertation (cf. Section 2.2). In particular, using the techniques in Section 2.2, it can be shown that the advantage of a ("generic") adversary is at most $O\left(q^2 2^\lambda / p\right)$, where $|\mathbb{G}| = p$, q is the sum of the number of group oracle queries and the number $(n+1)$ of pairs $(t_i, \alpha \cdot t_i^{-1})$, and λ is a bound on the amount of information obtained on each t_i and $\alpha \cdot t_i^{-1}$, measured in bits.

2.1.2 The Conjecture

By $\mathrm{MSB}_{k,p}(z)$, we mean the (integer representing the) k most significant bits of $z \pmod p$, where the elements of $\mathbb{Z}_p$ are represented by integers of fixed bit-length $m = \lfloor \log p \rfloor + 1$. For instance, $\mathrm{MSB}_{2,7}(3) = 1$. The notation "log" always refers to logarithm to the base 2.

First we shall briefly recollect the notion of a *key encapsulation mechanism* (KEM) and a *stateful* KEM. Next we describe the ElGamal-based stateful KEM of [KP10a]. Then we discuss the conjecture in [KP10a] regarding the security of the scheme, and its relation to HS-HNP.

2.1.2.1 KEM and the Leakage Model

A key encapsulation mechanism KEM is a public-key cryptosystem used to establish a session key for a symmetric-key encryption scheme. Formally, KEM consists of three algorithms KG, Enc and Dec. The key generation algorithm KG on input a security parameter κ produces a pair of public and secret keys (pk, sk). The encapsulation algorithm Enc, taking only pk as input, outputs an encryption C of a key K. The decapsulation algorithm Dec on inputs sk and C outputs K. The goal of an adversary is to distinguish the encryption of a given key from that of a random key. An adversary may be a Chosen Plaintext Attack- (CPA-) adversary if it has no access to a decryption oracle. If it does have access to such an oracle then it is called a Chosen Ciphertext Attack (CCA) adversary. CCA adversaries can be further classified into CCA1- or CCA2-adversaries. A CCA1-adversary cannot query the decapsulation oracle after obtaining the challenge ciphertext.

In a KEM *with stateful decapsulation*, $\mathsf{KEM}^\star = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec1}, \mathsf{Dec2})$, the decapsulation algorithm Dec is split into two parts $\mathsf{Dec1}$ and $\mathsf{Dec2}$ executed consecutively. Each such algorithm uses different parts of the memory that therefore leak independent side-channel data, thus obeying to the *Only Computation Leaks/Split-State* model [MR04, LL12]. The secret key of $\mathsf{KEM}^\star$ now consists of two parts $sk_i = (\sigma_i, \sigma_i')$, each part residing on a different portion of the memory. $\mathsf{Dec1}$ can access only the part of the memory containing σ_i, while $\mathsf{Dec2}$ has access only to the other part containing σ_i'. The decapsulation procedures may update the secret key (to a functionally equivalent key)

after each access to it, say from sk_i to $sk_{i+1} = \left(\sigma_{i+1}, \sigma'_{i+1}\right)$. Note that in order to protect against continual leakage, the secret key must be "refreshed" regularly. We refer to one execution of the decapsulation query as a *round*.

The leakage in each round is modelled as the output of two adversarially chosen efficiently computable functions $f_i(\cdot)$ and $g_i(\cdot)$. The output of the two functions is bounded by λ bits each, where λ is the *leakage parameter*. The function $f_i(\sigma_{i-1}, r_i)$ models the leakage of one part of the memory containing σ_{i-1} and the internal randomness r_i used by $\mathsf{Dec1}$. Whereas, $g_i(\sigma'_{i-1}, w_i, r'_i)$ models the leakage of the other part of the memory containing σ'_{i-1}, the internal randomness r'_i used by $\mathsf{Dec2}$, and the information w_i shared between $\mathsf{Dec1}$ and $\mathsf{Dec2}$ to obtain the complete decapsulation key. The $\mathsf{KEM}^\star$ is said to be (κ, λ) secure under the Chosen Ciphertext with Leakage Attacks 1 (CCLA1) if the scheme remains secure even when an adversary can obtain λ bits of leakage from each of the two functions f_i and g_i, in every decapsulation query.

2.1.2.2 Stateful ElGamal KEM

The following well-known variant of the ElGamal KEM, where the secret key is multiplicatively shared, was studied in [KP10a].

Let the output of $\mathsf{Gen}(\kappa, \lambda)$ be a (multiplicatively written) cyclic group $\mathbb{G} = \langle g \rangle$ of prime order p, generated by g. It is required that $p-1$ has a large prime factor, as we shall see later. Let $\mathsf{EG}^\star = (\mathsf{KG}_{\mathsf{EG}^\star}, \mathsf{Enc}_{\mathsf{EG}^\star}, \mathsf{Dec1}_{\mathsf{EG}^\star}, \mathsf{Dec2}_{\mathsf{EG}^\star})$ be the stateful KEM defined in [KP10a, Section 3.1] as follows:

1. $\mathsf{KG}_{\mathsf{EG}^\star}(\kappa, \lambda)$: Compute $(\mathbb{G}, g, p) \leftarrow \mathsf{Gen}(\kappa, \lambda)$. Choose random $x \xleftarrow{\$} \mathbb{Z}_p$ and $\sigma_0 \xleftarrow{\$} \mathbb{Z}_p^*$. Set $h = g^x$ and $\sigma'_0 = x \cdot \sigma_0^{-1} \pmod p$. The public key is $pk = (\mathbb{G}, g, p, h)$, and the secret key is $sk = (\sigma_0, \sigma'_0)$.
2. $\mathsf{Enc}_{\mathsf{EG}^\star}()$: Choose random $l \xleftarrow{\$} \mathbb{Z}_p$. The ciphertext is $C = g^l$, and the key is $K = h^l$.
3. $\mathsf{Dec1}_{\mathsf{EG}^\star}(\sigma_{i-1}, C)$: Choose random $r_i \xleftarrow{\$} \mathbb{Z}_p^*$. Set $\sigma_i \leftarrow \sigma_{i-1} \cdot r_i \pmod p$, and $K'_i = C^{\sigma_i}$. Return (r_i, K'_i).
4. $\mathsf{Dec2}_{\mathsf{EG}^\star}\left(\sigma'_{i-1}, (r_i, K'_i)\right)$: Set $\sigma'_i \leftarrow \sigma'_{i-1} \cdot r_i^{-1} \pmod p$, and $K = K_i'^{\sigma'_i}$. Return K as the shared secret key.

Claim [KP10a, Conjecture 1] $\mathsf{EG}^\star$ is CCLA1 secure if $p-1$ has a large prime factor.

The above statement is incomplete if the leakage parameter λ (i.e., the amount of leakage from each of $\mathsf{Dec1}_{\mathsf{EG}^\star}$ and $\mathsf{Dec2}_{\mathsf{EG}^\star}$) is not specified. For instance, the conjecture could hold for $\lambda \ll \frac{\log p}{2}$. More precisely, $\lambda < \frac{\log p}{2} - \omega(\log \log p)$ to make it infeasible to guess the remaining bits by a brute force attack. Since leakage is modelled by functions $f_i(\sigma_{i-1}, r_i)$ and $g_i\left(\sigma'_{i-1}, (r_i, K'_i)\right)$, it is easy see that if $\lambda > \log p/2$, then one can completely recover r_i in the i^{th} round

and also some bits of the initial state (σ_0, σ'_0). In at most $2\lceil \log p \rceil$ rounds, the secret key x can be fully recovered. This is the trivial attack. It is also necessary that the Decisional Diffie-Hellman problem is hard in the group $\mathbb{G}$.

The reason why $p-1$ must have a large prime factor is due to the fact that it is easy to compute discrete logarithms in $\mathbb{Z}_p^*$ otherwise. Since $\log(\sigma_i \cdot \sigma'_i) = \log \sigma_i + \log \sigma'_i$, one can effectively transform the multiplicative sharing of the secret key into additive sharing if it is feasible to extract discrete logarithms. In [KP10a, Section 1.1], an attack is outlined on schemes using additive sharing, where $x = \tau_i + \tau'_i \pmod p$. The attack requires only a few bits of leakage on τ_i and τ'_i in each round i. Since $\tau_i + \tau'_i$ can only be equal to either x or $x+p$, we can compute few bits of x and/or $x+p$ at a time from the corresponding bits of τ_i and τ'_i (and carries, if any). This shows that the Generic HS-HNP (Definition 2.4) is easy to solve in the group $(\mathbb{Z}_p, +)$ with only a few bits of leakage from each share. Note that this attack will not apply for the multiplicative sharing since $\sigma_i \cdot \sigma'_i = x + s \cdot p$, for many possible values of s $(0 \le s \le p-2)$.

2.1.2.3 Relationship to the HS-HNP

In the following we argue that it is possible to obtain 2λ most significant bits of each of the two shares of the secret key in $\mathsf{EG}^\star$. This is a consequence of the fact that λ bits of σ_i (respectively, σ'_i) can be leaked from each of $f_i(\sigma_{i-1}, r_i)$ and $f_{i+1}(\sigma_i, r_{i+1})$ (respectively, $g_i\left(\sigma'_{i-1}, (r_i, K'_i)\right)$ and $g_{i+1}\left(\sigma_i, \left(r_{i+1}, K'_{i+1}\right)\right)$), where $i \ge 1$. After $2n+2$ rounds of execution of $(\mathsf{Dec1}_{\mathsf{EG}^\star}, \mathsf{Dec2}_{\mathsf{EG}^\star})$, an adversary will be able to obtain $n+1$ pairs $\left(\mathrm{MSB}_{2\lambda,p}\left(\sigma_{2i+1}\right), \mathrm{MSB}_{2\lambda,p}\left(\sigma'_{2i+1}\right)\right)$ of the secret key, where $i = 0, \ldots, n$.

The above observation leads us to an instance of HS-HNP (Definition 2.3) with $\alpha = x$, $t_i = \sigma_{2i+1}$ and $\frac{\alpha}{t_i} \equiv \sigma'_{2i+1} \pmod p$, where $i = 0, \ldots, n$. Note that t_i is uniform random and independent in $\mathbb{Z}_p^*$. Hence investigating HS-HNP is a natural approach to resolve the above conjecture. This is the topic of the next section.

Let us notice that in [KP10b] it is wrongly stated that breaking the CCLA1 security of the stateful ElGamal KEM is related to the Hidden Multipliers - HNP (Definition 2.2). That is, using our notation, the HM-HNP boils down to an adversary that is getting leakage on t_i and αt_i, which corresponds to $t_i = \sigma_{2i+1}$ and $\alpha t_i \equiv \sigma'_{2i+1} t_i^2 \pmod p$, where $i = 0, \ldots, n$. Apparently, the value αt_i is never computed in the ElGamal KEM nor can possibly be computed by leakage functions.

2.1.3 Hidden Shares - Hidden Number Problem

Let p be an m-bit integer. We have $m = \lfloor \log p \rfloor + 1$. Let $y_i = 2^{m-k} \cdot \mathrm{MSB}_{k,p}(t_i)$ and $b_i = 2^{m-k} \cdot \mathrm{MSB}_{k,p}\left(\frac{\alpha}{t_i}\right)$ in Definition 2.3. Let

$$t_i = y_i + \delta_i,$$
$$\frac{\alpha}{t_i} \pmod p = b_i + \epsilon_i,$$

where $0 \leq \delta_i, \epsilon_i < 2^{m-k}$, where $i = 0, \ldots, n$. Note that the integers y_i and b_i are known, while the integers δ_i and ϵ_i are unknown. We have

$$(y_i + \delta_i)(b_i + \epsilon_i) \equiv \alpha \pmod p. \tag{2.1}$$

Since α is an unbounded variable, we will eliminate it from the $n+1$ equations represented by (2.1). We obtain

$$(y_i + \delta_i)(b_i + \epsilon_i) - (y_0 + \delta_0)(b_0 + \epsilon_0) \equiv 0 \pmod p.$$

On rearranging the terms, we obtain a set of n equations (for $i = 1, \ldots, n$) as

$$\begin{aligned}(-1)\,\delta_0\epsilon_0 + (1)\,\delta_i\epsilon_i + (-b_0)\,\delta_0 + (b_i)\,\delta_i + (-y_0)\,\epsilon_0 + (y_i)\,\epsilon_i \\ + (y_i b_i - y_0 b_0) \quad \equiv \quad 0 \pmod p.\end{aligned} \tag{2.2}$$

For the sake of clarity, let us denote the coefficients in the above $\left(i^{\text{th}}\right)$ relation by A_i, B_i, C_i, D_i, E_i, F_i, G_i, respectively in the order. Equation (2.2) can now be rewritten as

$$A_i\delta_0\epsilon_0 + B_i\delta_i\epsilon_i + C_i\delta_0 + D_i\delta_i + E_i\epsilon_0 + F_i\epsilon_i + G_i \equiv 0 \pmod p. \tag{2.3}$$

Note again that the only unknowns in the above equation are δ_0, δ_i, ϵ_0 and ϵ_i. Equation (2.3) can be rewritten over the integers as

$$A_i\delta_0\epsilon_0 + B_i\delta_i\epsilon_i + C_i\delta_0 + D_i\delta_i + E_i\epsilon_0 + F_i\epsilon_i + G_i + p \cdot \mu_i = 0, \tag{2.4}$$

where the μ_i are unknowns, and $i = 1, \ldots, n$. The quantities μ_i are of little interest compared to that of δ_i and ϵ_i. We shall now construct a lattice that captures the relations defined by (2.4). Since (2.4) contains non-linear terms like $\delta_0\epsilon_0$ and $\delta_i\epsilon_i$, we "linearize" the relation by treating the non-linear terms as a separate variable. It is also desirable to have the solution we are looking for correspond to a "short" vector in the lattice. Our construction is similar to the one in [BHHG01, Section 3.1].

2.1.3.1 Setting up the Lattice

The lattice we construct has dimension $4n + 4$ and it is represented by a $(4n + 4) \times (4n + 4)$ matrix M consisting of rational entries. The lattice is generated as the row span of the matrix M. The structure of M is as follows:

$$M = \begin{pmatrix} J & R \\ 0 & P \end{pmatrix}, \tag{2.5}$$

where J and P are diagonal matrices having dimensions $(3n+4) \times (3n+4)$ and $n \times n$, respectively. Matrix R has dimensions $(3n+4) \times n$. The rows of M correspond to the terms present in the n relations represented by (2.4). The first row is associated with the constant term, the next $n+1$ rows correspond to the variables δ_i, next $n+1$ rows with ϵ_i, while the further $n+1$ rows correspond to $\delta_i \epsilon_i$. The last n rows are associated with the terms μ_i. Each of the last n columns of M correspond to a relation in (2.4), while the first $3n+4$ columns are associated with the inverse of an upper bound on the size of the quantities 1, δ_i, ϵ_i and $\delta_i \epsilon_i$ (in the solution we are interested in). In what follows, we give a complete description of the matrix M.

Let $P[i', j']$ denote the entry in the i'^{th} row and the j'^{th} column of the matrix P. The matrix P has p on all of its main diagonal, i.e. $P[i,i] = p$ for $1 \leq i \leq n$. The i^{th} row of P corresponds to the term μ_i in (2.4). The diagonal matrix J has $J[1,1] = 1$ (for the constant term), $J[i', i'] = 2^{k-m}$ for $2 \leq i' \leq 2n+3$ (for terms δ_i and ϵ_i), and $J[i', i'] = 2^{2(k-m)}$ for $2n+4 \leq i' \leq 3n+4$ (for terms $\delta_i \epsilon_i$). As we shall later see, these entries of J are required to bound the norm of the vector corresponding to our solution. Each of the n relations of (2.4) is described in matrix R (excluding terms $p \cdot \mu_i$, which are described by matrix P). The entry $R[i', j']$ is the coefficient of the term corresponding to row i' in the j'^{th} relation. Hence the columns of matrices R and P together completely describe the system of equations (2.4).

As an illustration, the matrix M is described for the case $n = 2$ in Figure 2.1, where $\phi = 2^{k-m}$. The terms corresponding to the 12 rows of M are (from top to bottom) 1, δ_0, δ_1, δ_2, ϵ_0, ϵ_1, ϵ_2, $\delta_0\epsilon_0$, $\delta_1\epsilon_1$, $\delta_2\epsilon_2$, μ_1, and μ_2, respectively.

$$M = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & G_1 & G_2 \\
0 & \phi & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & C_1 & C_2 \\
0 & 0 & \phi & 0 & 0 & 0 & 0 & 0 & 0 & 0 & D_1 & 0 \\
0 & 0 & 0 & \phi & 0 & 0 & 0 & 0 & 0 & 0 & 0 & D_2 \\
0 & 0 & 0 & 0 & \phi & 0 & 0 & 0 & 0 & 0 & E_1 & E_2 \\
0 & 0 & 0 & 0 & 0 & \phi & 0 & 0 & 0 & 0 & F_1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \phi & 0 & 0 & 0 & 0 & F_2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \phi^2 & 0 & 0 & A_1 & A_2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \phi^2 & 0 & B_1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \phi^2 & 0 & B_2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p
\end{pmatrix}$$

Figure 2.1: The matrix M for the case $n = 2$. Let $\phi = 2^{k-m}$.

Let $\epsilon_i = e_i$, $\delta_i = d_i$ $(0 \leq i \leq n)$ and $\mu_i = u_i$ $(1 \leq i \leq n)$ be a solution to the system of equations (2.4), where $0 \leq e_i, d_i < 2^{m-k}$. Note that such a solution exists from the way the system was constructed. Let v be a (row) vector, of

length $4n + 4$, defined as follows:

$$v = \langle 1,\, d_0,\, \ldots,\, d_n,\, e_0,\, \ldots,\, e_n,\, d_0e_0,\, \ldots,\, d_ne_n,\, u_1,\, \ldots,\, u_n \rangle\,.$$

Since e_i, d_i and u_i satisfy the system (2.4), it is easy to see that

$$v \cdot M = \left\langle 1, \frac{d_0}{2^{m-k}}, \ldots \frac{d_n}{2^{m-k}}, \frac{e_0}{2^{m-k}}, \ldots \frac{e_n}{2^{m-k}}, \frac{d_0e_0}{2^{2(m-k)}}, \ldots \frac{d_ne_n}{2^{2(m-k)}}, 0, \ldots 0 \right\rangle. \tag{2.6}$$

Note that the vector $v \cdot M$ has a leading 1, and n trailing zeros. Its length is $4n + 4$ and its Euclidean norm $\|v \cdot M\|_2$ satisfies

$$\|v \cdot M\|_2 < \sqrt{3n + 4}. \tag{2.7}$$

If we are able to find the vector $v \cdot M$, then we can readily recover the values d_i, e_i, and hence solve the system of (2.4). Since the vector $v \cdot M$ has a bounded length, we can try to choose a value for k such that the resulting lattice has a sufficiently large determinant, and hence is unlikely to have many vectors shorter than $\|v \cdot M\|_2$. Then we can run a lattice reduction algorithm, say LLL [LLL82], to obtain reduced basis vectors. We then hope that by exploiting the structure of the vector $v \cdot M$, we can obtain it as a simple combination of a few short basis vectors. Since it appears hard to rigorously bound the probability of failure, we had to content ourselves with the heuristic arguments, as it is common with these techniques.

We have done an implementation of our method to justify the heuristics, and to get an estimate of the running time. The results have been presented in Section 2.1.3.3.

2.1.3.2 Estimating k

We now give an estimate for the suitable values of k. The Gaussian heuristic gives an estimate of the expected number of lattice points in a sphere of given volume. Consider a full rank lattice L' in $\mathbb{R}^{n'}$whose determinant is $\det(L')$. Let $V_{n'}(r')$ denote the volume of an n'-ball $S_{n'}(r')$ of radius r', centered at the origin.

Lemma 2.1 [Gaussian Heuristic [Ngu10, pp. 28]] The "expected" number of lattice points of L' in $S_{n'}$ is $\frac{V_{n'}(r')}{\det(L')}$.

An explicit formula for $V_{n'}(r')$ is

$$V_{n'}\left(r'\right) = \frac{\pi^{\frac{n'}{2}}}{\Gamma(\frac{n'}{2} + 1)} r'^{n'}, \tag{2.8}$$

where $\Gamma(\cdot)$ is the Gamma function.

In our case, the determinant of the lattice M in (2.5), $\det(M)$, is

$$\det(M) = \frac{p^n}{2^{(m-k)(4n+4)}} \geq \frac{2^{(m-1)n}}{2^{(m-k)(4n+4)}}. \tag{2.9}$$

The above inequality follows from the fact that p is an m-bit integer. Note that the value of $\det(M)$ increases with the value of k. By (2.7), $\|v \cdot M\|_2 < \sqrt{3n+4}$. We would like to choose such a value for k so that the expected number of lattice points of M in $S_{4n+4}(\sqrt{3n+4})$ is at most one. By Lemma 2.1 and (2.9), it suffices if

$$\frac{2^{(m-1)n}}{2^{(m-k)(4n+4)}} \geq V_{4n+4}(\sqrt{3n+4}).$$

On rearranging the above inequality, we get

$$\frac{k}{m} \geq \frac{3+\frac{4}{n}}{4+\frac{4}{n}} + \left(\frac{n + \log_2\left(V_{4n+4}(\sqrt{3n+4})\right)}{m\,(4n+4)} \right). \tag{2.10}$$

From (2.8), $V_{4n+4}(\sqrt{3n+4}) = \frac{\pi^{2n+2}}{(2n+2)!}\,(3n+4)^{\,2n+2}$. Therefore, in (2.10),

$$\frac{n + \log_2\left(V_{4n+4}(\sqrt{3n+4})\right)}{m\,(4n+4)} = O\left(\frac{\log n}{m}\right).$$

If $1 \ll n \ll m$, we obtain

$$k = \left(\frac{3}{4} + o\,(1)\right) m. \tag{2.11}$$

The following remark summarizes the above result.

Remark 2.1 There is an efficient (heuristic) method to solve the HS-HNP (Definition 2.3) with $k = \left(\frac{3}{4} + o\,(1)\right) m$.

On applying the above method to attack the scheme $\mathsf{KEM}_{\mathsf{EG}}$ (c.f. Section 2.1.2.3), we obtain the following remark that disproves the claim in Section 2.1.2.2. Note that $k = 2\lambda$.

Remark 2.2 On applying the above method to attack the scheme $\mathsf{EG}^\star$ (c.f. Section 2.1.2.3), we obtain that the stateful KEM $\mathsf{EG}^\star$ is *not* CCLA1 secure if the leakage parameter $\lambda \geq \left(\frac{3}{8} + o\,(1)\right) \log p$. We would like to note that our attack does not exploit any information about the elements of the underlying group $\mathbb{G}$ in $\mathsf{EG}^\star$. Hence this attack will work for any instantiation of the group. Also note that the attack can be easily adapted to composite-order groups.

2.1.3.3 Implementation Details

We have implemented the above method to solve HS-HNP. We have used the "PARI/GP" computer algebra system [The12] in our implementation. The experiments were run on an Intel(R) Core i7-2600 CPU with 4 GB RAM, running cygwin (ix86/GMP-4.2.1 kernel) 32-bit version. The results are given in Table 2.1. For every (n, m) pair, 10 random m-bit primes p were chosen. For each p, 10 random hidden numbers α were chosen. The running time reported is averaged over 100 (p, α) pairs for each row of the table. For lattice reduction, we have used the routine `qflll` in PARI/GP. The source code used for the experiments is given in Section 2.1.4.

n	m (bits)	k (bits)	$\frac{k}{m}$	dimension(M)	time (sec)
2	256	216	0.844	12	0.031
2	512	429	0.838	12	0.087
2	1024	856	0.836	12	0.290
5	256	205	0.800	24	0.507
5	512	408	0.797	24	1.367
5	1024	813	0.794	24	4.468
10	256	200	0.781	44	4.144
10	512	398	0.777	44	10.911
10	1024	794	0.775	44	32.395

Table 2.1: HS-HNP: implementation results

In the experiments, we have observed that there is one (reduced) basis vector of very low norm, of order $\frac{1}{2^{m-k}}$. This is because the rows of the matrix M (Equation (2.5)) corresponding to the terms $\delta_i \epsilon_i$ $(0 \leq i \leq n)$ can add up to produce the vector

$$\left\langle \underbrace{0, \ldots, 0}_{2n+3}, \underbrace{2^{k-m}, \ldots, 2^{k-m}}_{n+1}, \underbrace{0, \ldots, 0}_{n} \right\rangle.$$

For instance, in Figure 2.1, we have $A_1 = A_2 = -1$, and $B_1 = B_2 = 1$. Hence by adding up the corresponding three rows, we obtain the vector

$$\left\langle 0, 0, 0, 0, 0, 0, 0, 2^{k-m}, 2^{k-m}, 2^{k-m}, 0, 0, 0 \right\rangle.$$

We have also observed that there are many (reduced) basis vectors with norm of about $\sqrt{3n+4}$. In order to overcome these issues, we "randomize" the relations from Equation (2.2) by multiplying each relation by a random (independent) non-zero element of $\mathbb{Z}_p$. Of course, this will not alter the solution set. In spite of doing so, there will be very short vectors of the order $\frac{1}{2^{m-k}}$. But the experiments suggest that there will be only one (reduced) basis vector (second shortest) of norm about $\sqrt{3n+4}$, and we can get the required values

of d_i and e_i from the corresponding entries of the basis vector. This heuristic has *not* failed even once for the 900 (p, α) pairs of Table 2.1, provided k is chosen according to Equation (2.10).

2.1.4 PARI/GP script for solving HS-HNP

```
\\ Input p - m-bit prime.
\\ Input y,b - vector of (k-)MSBs, of length n+1.

HSHNP(n,m,k,p,y,b) =
{
local(M,R,S,T,i,j,tmp,ans);

M = matrix(4*n+4,4*n+4);
R = matrix(3*n+4,n);

\\  Matrix J

for(i=1, 3*n+4,
  if(i==1, M[i,i]=1 );
  if(i>1 && i<=(2*n+3), M[i,i]=2^(k-m) );
  if(i>(2*n+3) && i<=(3*n+4), M[i,i]=2^(2*(k-m)) );
  );

\\  Matrix P

for(i=1,n, M[i+(3*n+4),i+(3*n+4)]=p);

\\  Matrix R, M

for(j=1, n,
  while((tmp = random(p))==0,);
  for(i=1, 3*n+4,
    if(i==1, R[i,j]=((y[j+1]*b[j+1]-y[1]*b[1])*tmp)%p);
    if(i==2, R[i,j]=(-1*b[1]*tmp)%p);
    if(i>2 && i<=(n+2) && j==(i-2), R[i,j]=(b[j+1]*tmp)%p);
    if(i==(n+3), R[i,j]=(-1*y[1]*tmp)%p);
    if(i>(n+3) && i<=(2*n+3) && j==(i-(n+3)),
        R[i,j]=(y[j+1]*tmp)%p;
       );
    if(i==(2*n+4), R[i,j]=(-1*tmp)%p);
    if(i>(2*n+4) && i<=(3*n+4) && j==(i-(2*n+4)),
        R[i,j]=(1*tmp)%p
       );
    M[j+(3*n+4),i]=R[i,j];      \\Transpose of "M"
    );
  );

T = qflll(M);
S = M*T;        \\Reduced basis

\\compute the hidden number

tmp = ((y[1]+2^(m-k)*abs(S[2,2])));
```

```
49 ans = tmp*(b[1]+2^(m-k)*abs(S[n+3,2])))%p;
50
51 ans \\return
52 }
```

2.2 Split-State Bilinear ElGamal KEM

2.2.1 Introduction

In Section 2.1.1, we briefly mentioned about a leakage-resilient key encapsulation mechanism proposed by Kiltz and Pietrzak [KP10a]. It is a pairing-based stateful variant of the ElGamal encryption scheme (called BEG-KEM), where the secret key is an element of the pairing base group (essentially a point in the group of points of an elliptic curve). The secret key is divided into two shares, which are re-shared at each new decryption call by using multiplicative blinding. To decrypt, one takes the first half of the secret key, refreshes it, and uses it as the input to a pairing calculation. In the second step, the second half of the secret key is updated with the blinding used for refreshing; it is then used as the input to a new pairing calculation; and finally the two pairing values are multiplied to obtain a decapsulated symmetric key (for details see Section 2.2.2). This is one of the very few schemes admitting continual leakage (maybe the only one?) that one could dare to implement on an embedded processor, for instance on a smart phone.

The result proven in [KP10a], which holds under a variant of the generic group model tailored to pairing groups uses a bounded leakage assumption. Roughly speaking, it is required that the *data leaked* against side-channel attacks conform to the split-state leakage model (hence also satisfy the only computation leaks information axiom), shall be significantly smaller than κ for a single measurement, where κ is the security parameter (e.g. $\kappa = 128$). These leakages are modeled as an oracle that answers values $f(\cdot)$ for adaptively chosen arbitrary (but efficiently computable) functions f on input the secret data being used in the calculation. This kind of requirement, that may look reasonable for a theoretician used to study cryptographic primitives in the so-called *black-box model* might seem completely unrealistic to the practitioner (though we could, perhaps not unreasonably, argue that the amount of "useful" information leaked is bounded). An an example, let us recall the figure gathered in [SPY13], where it is pointed out that the leaking of a block cipher recently reported in [Mor12], consisted of 200000 traces leading to more than 1.5 Gigabits of data storage.

2.2.1.1 Our Contribution

In this chapter, we analyze, modify, implement and evaluate BEG-KEM. We start our investigation by proposing and testing a relaxation on the requirement of 'bounded leakage size' in the split-state model. We weaken the restriction on the image size of the leakage functions in these models to asking that the random variables used to refresh the secret key shall have enough min-entropy left given the leakage, with no limitation on the 'size' of this leakage. This is an altogether more reasonable leakage bound assumption,

which could eventually be met by clever implementations (in fact we provide an implementation candidate). We give a new security reduction using the generic bilinear group axiom for BEG-KEM in this relaxed leakage model, which turns out to be tighter than the original reduction in [KP10a] in the split-state model.

Secondly, we observe that the blinding mechanism originally proposed is susceptible to invalidate the leakage bound assumption. This is because to perform blinding, one computes an exponentiation G^{r_i} for a random integer r_i, which if implemented in a naive way, can almost completely leak r_i, even with a simple power analysis attack (i.e. with a single power trace), as we discuss in Section 2.2.5. The authors in [KP10a] did not discuss how exponentiation shall be implemented to meet the leakage bound, nor we can currently find a exponentiation algorithm with these guarantees. Thus, their positive result risks to be void.

This is why we propose, as our third contribution, an advanced BEG-KEM+, where we avoid blinding by an exponentiation G^{r_i} for a random integer r_i. Our modification is based on the observation that the knowledge of the exponent r_i is not needed to perform a successful decryption, but it suffices to build a *random element* in a suitable pairing base group. We propose instead to use a random encoding into asymmetric pairing groups by Fouque and Tibouchi [FT12]. It turns out that this encoding produces a random element in the base group, and can naturally be implemented in such a way that the leakage expected against a single measurement is arguably minimal (see Section 2.2.5).

Fourthly, we report the implementation of BEG-KEM+ in ANSI C on an ARM based microcontroller. BEG-KEM+ is, to our knowledge, the first implementation and evaluation of a public-key scheme from the leakage-resilient literature. We also assess its (theoretical) resistance against power analysis attacks from a practical perspective, taking into account the state-of-the-art in side-channel cryptanalysis.

2.2.2 Stateful Bilinear ElGamal KEM

In this section, we first recall the basics of the notion of min-entropy, bilinear groups, and the generic bilinear group model. Then we introduce the concept of stateful KEM and the security under non-adaptive chosen-ciphertext attacks in the presence of continual min-entropy leakage (CCmLA1). We note again that the class of leakage functions allowed in our model (based on lowering min-entropy) is broader than the bounded length model (CCLA1) used in [KP10a]. Next, we recollect the stateful bilinear ElGamal KEM construction from [KP10a] and present a new (and tighter) security reduction in our relaxed leakage model.[1]

[1] We point out that the authors of [KP10a] mention that their results also carry over to a relaxed leakage model, close in spirit to ours. However this model is not fully detailed, and additionally no justification of this fact is given in [KP10a] nor in [KP10c].

2.2.2.1 Min-Entropy

Let X be a finite random variable with probability distribution Pr. The *min-entropy* of X, denoted $\mathbf{H}_\infty(X)$, is defined as $\mathbf{H}_\infty(X) := -\log_2\big(\max_x \Pr[X = x]\big)$. Min-entropy is a standard measure of the worst-case predictability of a random variable. Let Z be a random variable. The *average conditional min-entropy* of X given Z, denoted $\tilde{\mathbf{H}}_\infty(X \mid Z)$, is defined as

$$\tilde{\mathbf{H}}_\infty(X \mid Z) := -\log_2\Big(\mathop{\mathbb{E}}_{z \leftarrow Z}\Big[\max_x \Pr[X = x \mid Z = z]\Big]\Big).$$

Average conditional min-entropy is a measure of the worst-case predictability of a random variable given a correlated random variable.

Lemma 2.1. [[DORS08]] *Let $f : X \to \{0,1\}^{\lambda'}$ be a function on X. Then $\tilde{\mathbf{H}}_\infty(X \mid f(X)) \geq \mathbf{H}_\infty(X) - \lambda'$.*

Lemma 2.2. [Schwartz-Zippel [Zip79, Sch80]] *Let $\mathbb{F}$ be an arbitrary finite field. Let $\mathsf{Q} \in \mathbb{F}[\mathsf{X}_1, \ldots, \mathsf{X}_k]$ be a non-zero multivariate polynomial of (total) degree at most d. Then the number of zeroes of Q (over $\mathbb{F}$) is at most $d \cdot |\mathbb{F}|^{k-1}$.*

We propose the following variant of the Schwartz-Zippel Lemma for prime fields, which extends to arbitrary finite fields in a straightforward manner. [Sch80, Zip79]. This result of ours first appeared in [GV12].

Lemma 2.3. [Schwartz-Zippel; min-entropy version] *Let $\mathsf{F} \in \mathbb{Z}_q[\mathsf{X}_1, \ldots, \mathsf{X}_n]$ be a non-zero polynomial of (total) degree at most d. Let P_i $(i = 1, \ldots, n)$ be probability distributions on $\mathbb{Z}_q$ such that $\mathbf{H}_\infty(P_i) \geq \log q - \lambda'$, where $0 \leq \lambda' \leq \log q$. If $x_i \overset{P_i}{\leftarrow} \mathbb{Z}_q$ $(i = 1, \ldots, n)$ are independent, then $\Pr[\mathsf{F}(x_1, \ldots, x_n) = 0] \leq 2^{\lambda'} \dfrac{d}{q}$.*

Proof. We prove the result by induction. When $n = 1$, the univariate polynomial F has at most d roots. Since $\mathbf{H}_\infty(P_1) \geq \log q - \lambda'$, we have $\Pr[\mathsf{F}(x_1) = 0] \leq d\, 2^{-(\log q - \lambda')} = \frac{d}{q}\, 2^{\lambda'}$.

Let us now prove the result for the n-variables case assuming the result for the $(n-1)$-variables case. On writing F as a polynomial in X_1 with coefficients in $\mathbb{Z}_q[\mathsf{X}_2, \ldots, \mathsf{X}_n]$, let i $(i \geq 1)$ be the degree of X_1 in the leading term and $\mathsf{F}' \in \mathbb{Z}_q[\mathsf{X}_2, \ldots, \mathsf{X}_n]$ be the leading coefficient. The probability

$$\begin{aligned}\Pr[\mathsf{F}(x_1, \ldots, x_n) = 0] \leq \Pr[\mathsf{F}(x_1, \ldots, x_n) = 0 \mid \mathsf{F}'(x_2, \ldots, x_n) \neq 0] \\ + \Pr[\mathsf{F}'(x_2, \ldots, x_n) = 0].\end{aligned}$$

F' is now a non-zero polynomial, of degree at most $d-i$, in only $n-1$ variables. By induction hypothesis, we have $\Pr[\mathsf{F}'(x_2, \ldots, x_n) = 0] \leq \frac{d-i}{q}\, 2^{\lambda'}$. When $\mathsf{F}'(x_2, \ldots, x_n) \neq 0$, we have $\Pr[\mathsf{F}(x_1, \ldots, x_n) = 0] \leq \frac{i}{q}\, 2^{\lambda'}$ because degree of F in X_1 is i $(i \geq 1)$ and the distributions P_i $(i = 1, \ldots, n)$ are independent. Hence $\Pr[\mathsf{F}(x_1, \ldots, x_n) = 0] \leq \frac{d}{q}\, 2^{\lambda'}$. Note that the parameter n does not appear in the above bound. □

Corollary 2.1. *If* $\lambda' < \log q - \omega(\log\log q)$ *in Lemma 2.3, then* $\Pr[\mathsf{F}(x_1, \ldots, x_n) = 0]$ *is negligible (in* $\log q$*).*

2.2.2.2 Bilinear Groups

Let κ denote the *s*ecurity parameter and λ denote the *leakage parameter*. Let $\mathsf{BGen}'(\kappa, \lambda)$ be a probabilistic bilinear group generator that outputs $(\mathbb{G}, \mathbb{G}_T, q, e', g)$ such that:

1. $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T$ are (multiplicatively written) cyclic groups of prime order q with binary operations $\cdot$ and $\star$, respectively. The size of q is κ bits.
2. $e' : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a map that is:
   (a) bilinear: $\forall u, v \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}$, $e'(u^a, v^b) = e'(u, v)^{ab}$.
   (b) non-degenerate: $e'(g, g) \neq 1$.

Such a group $\mathbb{G}$ is said to be a (symmetric) bilinear group if the above properties hold and the group operations in $\mathbb{G}$ and $\mathbb{G}_T$, and the map e' are efficiently computable. The group $\mathbb{G}$ is called as *base group* and $\mathbb{G}_T$ as *target group*. In the asymmetric setting there will two different base groups instead of one.

2.2.2.3 Generic Bilinear Group Model

The generic bilinear group (GBG) model [BBG05] is an extension of the generic group model [Sho97]. We use the notation used in [KP10a]. The encodings of the elements of $\mathbb{G}$ and $\mathbb{G}_T$ are given by random bijective maps $\xi : \mathbb{Z}_q \to \Xi$ and $\xi_T : \mathbb{Z}_q \to \Xi_T$, respectively, where Ξ and Ξ_T are sets of bit-strings. The group operations in $\mathbb{G}$ and $\mathbb{G}_T$, and evaluation of the bilinear map e are performed by three public oracles $\mathcal{O}$, $\mathcal{O}_T$ and $\mathcal{O}_e$, respectively, defined as follows. For all $a, b \in \mathbb{Z}_q$

- $\mathcal{O}(\xi(a), \xi(b)) := \xi(a + b \bmod q)$
- $\mathcal{O}_T(\xi_T(a), \xi_T(b)) := \xi_T(a + b \bmod q)$
- $\mathcal{O}_e(\xi(a), \xi(b)) := \xi_T(ab \bmod q)$

We assume that the (fixed) generator g of $\mathbb{G}$ satisfies $g = \xi(1)$, and also the (fixed) generator g_T of $\mathbb{G}_T$ satisfies $g_T = e(g, g) = \xi_T(1)$. The encoding of g is provided to all users of the group oracles. The users can thus efficiently sample random elements in both $\mathbb{G}$ and $\mathbb{G}_T$.

We further assume that $\Xi \cap \Xi_T = \phi$, $|\Xi| = |\Xi_T| = q$, and that the elements of Ξ and Ξ_T are from a well-defined set and are efficiently recognizable. For instance, the encodings in Ξ can comprise of the binary representation of the set $\{0, 1, \ldots, q-1\}$, where every string begins with '0' and all are of uniform length. The encodings in Ξ_T are similarly defined but instead begin with '1'.

Since the encodings are efficiently recognizable, the queries to a group oracle with an invalid encoding can be detected and an error can be raised. Such properties of encodings will be useful in the leakage setting. For simplicity, we assume that the users' queries to the oracles are all valid.

2.2.2.4 Stateful Key Encapsulation Mechanism

Formally, a split-state key encapsulation mechanism $\mathsf{KEM} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec1}, \mathsf{Dec2})$ consists of four polynomial-time algorithms. Recollect that κ denotes the *s*ecurity parameter, and that λ denotes the *leakage parameter*. The key generation procedure $\mathsf{KG}(\kappa, \lambda)$ takes as input κ and λ, and outputs the public key pk, a pair of initial (stateful) secret states (σ_0, σ'_0), and the public parameters $\mathbb{PP}$. The encapsulation procedure $\mathsf{Enc}(pk)$ takes as input pk, and outputs a secret symmetric key K and the corresponding ciphertext C. The stateful decapsulation procedure takes C as an input and outputs $K \in \mathcal{K}$. This procedure is split into two consecutive steps $\mathsf{Dec1}$ and $\mathsf{Dec2}$, where each step accesses distinct parts of the two secret states. The procedures $\mathsf{Dec1}$ and $\mathsf{Dec2}$ may also update the secret key using locally generated fresh randomness:

$$(\sigma_i, w_i) \xleftarrow{r_i} \mathsf{Dec1}(\sigma_{i-1}, C)\ ;\ (\sigma'_i, K) \xleftarrow{r'_i} \mathsf{Dec2}(\sigma'_{i-1}, w_i).$$

The scheme KEM is required to satisfy the following correctness property:

$$\begin{aligned}\Pr\big[\mathsf{Dec2}\left(\mathsf{Dec1}\left(\mathsf{Enc}\left(pk\right), \sigma_{i-1}\right) \backslash \sigma_i,\ \sigma'_{i-1}\right) = K\ :\\ \left(pk, \left(\sigma_{i-1}, \sigma'_{i-1}\right)\right) \leftarrow \left(\mathsf{KG}, \mathsf{Dec1}, \mathsf{Dec2}\right),\ K \leftarrow \mathsf{Enc}\left(pk\right)\big] \ &= \ 1.\end{aligned}$$

The security of the scheme KEM is defined by the following game:

KEM-CCmLA1$_{\mathsf{KEM}}(\mathcal{A}, \kappa, \lambda)$

$(pk, (\sigma_0, \sigma'_0)) \leftarrow \mathsf{KG}(\kappa, \lambda)$
$i := 1,\ w \leftarrow \mathcal{A}^{O^{\mathrm{CCmLA1}}(\cdot)}(pk)$
$b \xleftarrow{\$} \{0,1\}$
$(C, K_0) \leftarrow \mathsf{Enc}(pk)$
$K_1 \xleftarrow{\$} \mathcal{K}$
$b' \leftarrow \mathcal{A}(w, CK_b)$

KEM-Leak-Oracle $O^{\mathrm{CCmLA1}}(C, f_i, h_i)$

$(\sigma_i, w_i) \xleftarrow{r_i} \mathsf{Dec1}(\sigma_{i-1}, C)$
$(\sigma'_i, K) \xleftarrow{r'_i} \mathsf{Dec2}(\sigma'_{i-1}, w_i)$
$\Lambda_i := f_i(\sigma_{i-1}, r_i)$
$\Lambda'_i := h_i(\sigma'_{i-1}, r'_i, w_i)$
$i := i + 1$
Return $(K, \Lambda_i, \Lambda'_i)$

In the above experiment, $f_i(\sigma_{i-1}, r_i)$ and $h_i(\sigma'_{i-1}, r'_i, w_i)$ are (efficiently computable) leakage functions that the adversary can choose adaptively between the rounds. The functions $f_i(\cdot)$ and $h_i(\cdot)$ are such that the min-entropy of the individual inputs of the leakage functions is decreased by at most λ bits, given the corresponding leakages. More precisely, the requirement on the leakage functions is that

$$\tilde{\mathbf{H}}_\infty\left(\mathbf{t} \mid f_i(\sigma_{i-1}, r_i)\right) \geq \mathbf{H}_\infty\left(\mathbf{t}\right) - \lambda \quad \forall \mathbf{t} \ \in \ \sigma_{i-1} \cup r_i,$$

and

$$\tilde{\mathbf{H}}_\infty\left(\mathfrak{t} \mid h_i(\sigma'_{i-1}, r'_i, w_i)\right) \geq \mathbf{H}_\infty(\mathfrak{t}) - \lambda \quad \forall \mathfrak{t} \in \sigma'_{i-1} \cup r'_i \cup w_i.$$

Essentially, the above equations restrict the class of allowed leakage functions to those that do not decrease the min-entropy of each atomic parameter of the secret state by more than λ bits. For instance, if $w_i = \{w_{i,1}, w_{i,2}\}$, then we require that individually $w_{i,1}$ and $w_{i,2}$ have their min-entropy reduced by at most λ bits given the leakages.

Definition 2.1. [CCmLA1 security for KEM] *A key encapsulation mechanism* KEM *is* secure under non-adaptive chosen-ciphertext attacks in the presence of continual split-state leakage (CCmLA1)*, with min-entropy leakage bound* λ*, if* $\Pr[b' = b]$ *is at most negligibly greater than* $\frac{1}{2}$ *in the Experiment* $\text{KEM-CCmLA1}_{\mathsf{KEM}}(\mathcal{A}, \kappa, \lambda)$ *for any efficient adversary* $\mathcal{A}$*.*

Note that if in the above definition we would force the leakage functions to have output length of at most λ bits, then we would obtain the *CCLA1* security for KEM as defined in [KP10a]. From Lemma 2.1, we have that the conditional min-entropy of a random variable, given the leakage output of at most λ bits, cannot decrease by more than λ bits. Hence if a KEM is CCLA1 secure, then it is also CCmLA1 secure.

2.2.2.5 Bilinear ElGamal KEM

The scheme $\mathsf{BEG} = \left(\mathsf{KG}_{\mathsf{BEG}}, \mathsf{Enc}_{\mathsf{BEG}}, \mathsf{Dec1}_{\mathsf{BEG}}, \mathsf{Dec2}_{\mathsf{BEG}}\right)$ is as follows:

1. $\mathsf{KG}_{\mathsf{BEG}}(\kappa)$: Compute $\mathbb{PP} = (\mathbb{G}, \mathbb{G}_T, e', q, g) \leftarrow \mathsf{BGen}'(\kappa, \lambda)$ and randomly choose $x, t_0 \xleftarrow{\$} \mathbb{F}_q$. Set $X = g^x$, $\sigma_0 = g^{t_0}$, $\sigma'_0 = g^{x-t_0}$, and $X_T = e'(g,g)^x$. Return (pk, sk_0), where
   (a) the public key is $pk = (\mathbb{PP}, X_T)$.
   (b) the secret state is $sk_0 = (\sigma_0, \sigma'_0) \in \mathbb{G} \times \mathbb{G}$.
2. $\mathsf{Enc}_{\mathsf{BEG}}(pk)$*:* Choose a random $r \xleftarrow{\$} \mathbb{F}_q$. Compute the ciphertext $C = g^r$, and the derived key $K = X_T^r$. Return (C, K).
3. $\mathsf{Dec1}_{\mathsf{BEG}}(\sigma_{i-1}, C)$: Choose a random $t_i \xleftarrow{\$} \mathbb{F}_q$, set $\sigma_i = \sigma_{i-1} \cdot g^{t_i}$, $Y_i = e'(\sigma_i, C)$. Return (t_i, Y_i).
4. $\mathsf{Dec2}_{\mathsf{BEG}}(\sigma'_{i-1}, (t_i, Y_i), C)$: Set $\sigma'_i = \sigma'_{i-1} \cdot g^{-t_i}$, and $Y'_i = e'(\sigma'_i, C)$. Compute the derived key $K = Y_i \cdot Y'_i \in \mathbb{G}_T$. Return K.

The correctness of the scheme follows from the fact that $\sigma_i \cdot \sigma'_i = X \ \forall i \geq 0$ and using the bilinearity of $e'()$.

In [KP10a], the above scheme is shown to be secure in the generic bilinear group model [Sho97, BBG05] under (non-adaptive) chosen-ciphertext attacks in the presence of continual bounded-size leakage (in short, CCLA1 security).

The basic motivation for splitting the decapsulation step into two parts comes from the "only computation leaks information" axiom [MR04], which states that any leakage of information occurs only from the data that is being currently accessed by the computation.

Theorem 2.1. [KP10a, Theorem 1] *The scheme* BEG *(also called BEG-KEM) is CCLA1 secure in the generic bilinear group model. The advantage of an s-query adversary who gets at most λ bits of leakage per each invocation of* $\mathsf{Dec1}_{\mathsf{BEG}}$ *or* $\mathsf{Dec2}_{\mathsf{BEG}}$ *is at most* $\frac{s^3}{q}2^{2\lambda+1}$.

2.2.2.6 A CCmLA1 Security Reduction

We show that BEG-KEM is also leakage-resilient in the min-entropy leakage model introduced above, where leakage functions are not necessarily size-bounded. The only restriction is that the inputs to the leakage functions shall have enough min-entropy left, as a function of a leakage parameter λ, given the corresponding outputs. Interestingly, by using a different proof technique than [KP10c], we obtain a tighter bound on the adversarial CCLmA1 advantage than the bound claimed in [KP10a] for the adversarial CCLA1 advantage, w.r.t. the number of oracle queries s. In other words, with respect to the previous work, we provide here a new security reduction under a more realistic leakage model, and surprisingly we achieve better tightness.

Theorem 2.2. *The scheme* BEG-KEM *is CCmLA1 secure in the GBG model. The advantage of an s-query adversary with min-entropy leakage bound λ is* $\left(\frac{9s^2+3s}{q}\right)2^{2\lambda}$.

At a high level, the proof of this theorem proceeds in two steps. First we show in Theorem 2.3 that the scheme is secure if there is no leakage, i.e., CCA1 security. Note that the adversary is transparent to the internal details of secret state updates. Then, we complete the proof of CCmLA1 security by analyzing the effect of leakage on the CCA1 security.

The main idea to prove the CCA1 security is that the adversary will not be able to compute the derived symmetric key K_0 even after seeing the challenge ciphertext. To show this we just need to prove that K_0 cannot be written as a "linear combination" of the elements of $\mathbb{G}_T$ that it has got as input or can compute itself using the pairing oracle along with the input elements of $\mathbb{G}$. Hence in the GBG model it will not be able to distinguish the actual derived key or a randomly chosen key in $\mathbb{G}_T$. The challenger simulates the security game $\mathcal{G}$ to the adversary in the naive way. Also, the challenger simulates the generic bilinear group oracles in the usual way by maintaining lists of pairs of encodings and polynomials that represent the relation amongst group elements.

We then argue that that the proof for the non-leakage setting (i.e. proof of Theorem 2.3) and that for the leakage setting would be the same conditioned

on the fact that the adversary is unable to derive useful relation amongst the elements it has seen or guessed, and that it will not be able to compute and hence leak the full secret key X through the leakage functions, if λ is sufficiently small. Finally, we show that the probability of this event is increased by a factor of at most $2^{2\lambda}$ compared to the non-leakage setting. The formal proof is given below.

2.2.3 Proof of Theorem 2.2

The proof of this theorem proceeds in two steps. First we show in Theorem 2.3 that the scheme is secure if there is no leakage, i.e., CCA1 security. Note that the adversary is transparent to the internal details of secret state updates. Then, in Section 2.2.3.2, we complete the proof of CCmLA1 security by analyzing the effect of leakage on the CCA1 security.

2.2.3.1 Non-Leakage Setting: CCA1 Security

Theorem 2.3. *The scheme* BEG *is CCA1 secure in the generic bilinear group model, i.e., it is secure against non-adaptive chosen-ciphertext attacks if there is no leakage of the secret states. The advantage of an s-query adversary is at most $\frac{1}{2} + \frac{9s^2}{q}$.*

Proof. Let $\mathcal{A}$ be an s-query adversary that can break the CCA1 security of BEG. Hence $\mathcal{A}$ can make totally at most s group oracle, pairing oracle and decryption oracle queries. Let $s_{\mathcal{O}}$ denote the total number of calls to the oracles $\mathcal{O}$, $\mathcal{O}_T$ and $\mathcal{O}_e$, and s_D denote the number of calls to the decryption oracle O^{CCA1}. Thus $s_{\mathcal{O}} + s_D \leq s$. Let $\Pr^{\text{CCA1}}_{\mathcal{A},\mathsf{BEG}}$ denote the success probability of the adversary $\mathcal{A}$ in making a correct guess of b' in the security game. We show that

$$\Pr^{\text{CCA1}}_{\mathcal{A},\mathsf{BEG}} \leq \frac{1}{2} + \frac{9s^2}{q}.$$

for any s-query adversary $\mathcal{A}$ in the GBG model.

The main idea is to show that $\mathcal{A}$ will not be able to compute the derived symmetric key K_0 even after seeing the challenge ciphertext C^*. To show this we just need to prove that K_0 cannot be written as a "linear combination" of the elements of $\mathbb{G}_T$ that it has got as input or can compute itself using the pairing oracle $\mathcal{O}_e$ along with the input elements of $\mathbb{G}$. Hence in the GBG model it will not be able to distinguish the actual derived key or a randomly chosen key in $\mathbb{G}_T$. The challenger $\mathcal{C}$ simulates the security game $\mathcal{G}$ to $\mathcal{A}$ in the naive way. Also, $\mathcal{C}$ simulates the generic bilinear group oracles in the usual way by maintaining lists of pairs of encodings and polynomials that represent the relation amongst group elements.

We now formally describe the game $\mathcal{G}$. The description of the group oracles is typical for proofs in the generic group model (see [Sho97, Mau05, BB08]).

Description of Game $\mathcal{G}$: Let X, R,$\{\mathsf{U}_i : i \geq 1\}$ and $\{\mathsf{V}_i : i \geq 1\}$ be indeterminates. Intuitively, these (or other) polynomials represent the relation amongst the group elements that are output by a group oracle, or guessed by $\mathcal{A}$. The indeterminate X corresponds to the quantity x (discrete logarithm of the secret key), and R corresponds to the challenge ciphertext. Since $\mathcal{A}$ can query the group oracles with representations (from Ξ and Ξ_T) not previously obtained from the group oracles, in order to accommodate this case, we introduce the indeterminates U_i, V_i. The U_i correspond to the guessed elements of $\mathbb{G}$, whereas V_i correspond to the guessed elements of $\mathbb{G}_T$. We denote the lists $\{\mathsf{U}_i : i \geq 1\}$ and $\{\mathsf{V}_i : i \geq 1\}$ by $\{\mathsf{U}\}$ and $\{\mathsf{V}\}$, respectively.

$\mathcal{C}$ maintains two lists of pairs

$$\mathcal{L} = \{(\mathsf{F}_{1,i}\,,\, \xi_{1,i}) : 1 \leq i \leq \tau_1\}, \tag{2.12}$$

$$\mathcal{L}_T = \{(\mathsf{F}_{T,i}\,,\, \xi_{T,i}) : 1 \leq i \leq \tau_T\}. \tag{2.13}$$

The entries $\mathsf{F}_{1,i} \in \mathbb{Z}_q[\mathsf{X}, \mathsf{R}, \{\mathsf{U}\}]$, $\mathsf{F}_{T,i} \in \mathbb{Z}_q[\mathsf{X}, \mathsf{R}, \{\mathsf{U}\}, \{\mathsf{V}\}]$ are multivariate polynomials over $\mathbb{Z}_q$, whereas $\xi_{1,i}$, and $\xi_{T,i}$ are bit-strings in the encoding sets Ξ (of $\mathbb{G}$) and Ξ_T (of $\mathbb{G}_T$), respectively. The polynomials in lists $\mathcal{L}$ and $\mathcal{L}_T$ correspond to (more precisely, a superset of the) elements of $\mathbb{G}$ and $\mathbb{G}_T$, respectively, that $\mathcal{A}$ will ever be able to compute or guess. The values τ_1 and τ_T denote the respective list counters. In order to simplify the description, we view $\mathbb{Z}_q[\mathsf{X}, \mathsf{R}, \{\mathsf{U}\}]$ as a subring of $\mathbb{Z}_q[\mathsf{X}, \mathsf{R}, \{\mathsf{U}\}, \{\mathsf{V}\}]$.

Initially, $\tau_1 = 1$, $\tau_T = 1$, $\mathcal{L} = \{\ (1, \xi_{1,1})\ \}$, and $\mathcal{L}_T = \{\ (\mathsf{X}, \xi_{T,1})\ \}$. The bit-strings $\xi_{1,1}$, $\xi_{T,1}$ are set to random distinct strings from Ξ and Ξ_T, respectively. We assume that there is some ordering among the strings in the sets Ξ and Ξ_T (say, lexicographic ordering), so that given a string $\xi_{1,i}$ or $\xi_{T,i}$, it is possible to efficiently determine its index in the lists, if it exists. The initial state of the lists $\mathcal{L}$ and $\mathcal{L}_T$ correspond to the generator of $\mathbb{G}$ and the public key, respectively. The game begins by $\mathcal{C}$ providing $\mathcal{A}$ with the string $\xi_{1,1}$ from $\mathcal{L}$, and the string $\xi_{T,1}$ from $\mathcal{L}_T$.

Group Operation of $\mathbb{G}$: The calls made by $\mathcal{A}$ to the group oracle $\mathcal{O}$ are modeled as follows. For group operations in $\mathbb{G}$, $\mathcal{A}$ provides $\mathcal{C}$ with two operands (bit-strings) $\xi_{1,i}, \xi_{1,j}$ $(1 \leq i, j \leq \tau_1)$ in $\mathcal{L}$ and also specifies whether to multiply or divide them. $\mathcal{C}$ answers the query by first incrementing the counter $\tau_1 := \tau_1 + 1$, and computes the polynomial $\mathsf{F}_{1,\tau_1} := F_{1,i} \pm F_{1,j}$. If $\mathsf{F}_{1,\tau_1} = \mathsf{F}_{1,k}$ for some $k < \tau_1$, then $\mathcal{C}$ sets $\xi_{1,\tau_1} := \xi_{1,k}$. Otherwise, ξ_{1,τ_1} is set to a random string distinct from those already present in $\mathcal{L}$. The pair $(\mathsf{F}_{1,\tau_1}, \xi_{1,\tau_1})$ is appended to $\mathcal{L}$ and $\mathcal{C}$ provides $\mathcal{A}$ with ξ_{1,τ_1}. Note that the (total) degree of the polynomials $\mathsf{F}_{1,i}$ in $\mathcal{L}$ is at most one.

If $\mathcal{A}$ queries $\mathcal{O}$ with an encoding ξ not previously output by the oracle, then $\mathcal{A}$ increments the counter $\tau_1 := \tau_1 + 1$, sets $\xi_{1,\tau_1} := \xi$, and sets $\mathsf{F}_{1,\tau_1} := \mathsf{U}_{\tau_1}$. The pair $(\mathsf{F}_{1,\tau_1}, \xi_{1,\tau_1})$ is appended to $\mathcal{L}$. This step is carried out for each guessed operand.

Group Operation of $\mathbb{G}_T$: The group oracle $\mathcal{O}_T$ is modeled similar to $\mathcal{O}$, instead appropriately updating the counter τ_T, and appending the list $\mathcal{L}_T$ with the output $(\mathsf{F}_{T,\tau_T}, \xi_{T,\tau_T})$. $\mathcal{C}$ provides $\mathcal{A}$ with ξ_{T,τ_T}. For guessed operands in $\mathbb{G}_T$, a new variable V_{τ_T} is introduced instead.

Pairing Operation: For a pairing operation, $\mathcal{A}$ queries $\mathcal{C}$ with two operands $\xi_{1,i}, \xi_{1,j}$ $(1 \leq i, j \leq \tau_1)$ in $\mathcal{L}$. $\mathcal{C}$ first increments $\tau_T := \tau_T + 1$, and then computes the polynomial $\mathsf{F}_{T,\tau_T} := \mathsf{F}_{1,i} \cdot \mathsf{F}_{1,j}$. Again, if $\mathsf{F}_{T,\tau_T} = \mathsf{F}_{T,k}$ for some $k < \tau_T$, then $\mathcal{C}$ sets $\xi_{T,\tau_T} := \xi_{T,k}$. Otherwise, ξ_{T,τ_T} is set to a random string distinct from those already present in $\mathcal{L}_T$. The pair $(\mathsf{F}_{T,\tau_T}, \xi_{T,\tau_T})$ is appended to $\mathcal{L}_T$, and $\mathcal{C}$ provides $\mathcal{A}$ with ξ_{T,τ_T}. Note that the degree of the polynomials $\mathsf{F}_{T,i}$ in $\mathcal{L}_T$ is at most two.

Decryption: $\mathcal{C}$ answers decryption queries by $\mathcal{A}$ in the normal way by calling the pairing oracle $\mathcal{O}_e$, correspondingly updating the list $\mathcal{L}_T$, and by providing $\mathcal{A}$ the corresponding encoding in Ξ_T.

Challenge: $\mathcal{C}$ chooses a random bit $b \xleftarrow{\$} \{0, 1\}$. $\mathcal{C}$ adds the polynomial R and gives $\mathcal{A}$ the corresponding random distinct encoding in Ξ. If $b = 0$, $\mathcal{C}$ adds the polynomial XR to $\mathcal{L}_T$, else it adds a new polynomial V_{τ_T} (after incrementing τ_T) to $\mathcal{L}_T$. $\mathcal{A}$ is also given the corresponding encoding in Ξ_T.

End of Game $\mathcal{G}$**:** When $\mathcal{A}$ terminates it outputs a guess b' of b. Next, $\mathcal{C}$ chooses random values x, r, $\{u\}$, $\{v\}$, $\{r\} \leftarrow \mathbb{Z}_q$ for the indeterminates X, R, $\{\mathsf{U}\}$, $\{\mathsf{V}\}$, respectively. Then it evaluates the polynomials in lists $\mathcal{L}$ and $\mathcal{L}_T$.

Note that the adversary $\mathcal{A}$ will not be able to compute the polynomial XR from polynomials in $\mathcal{L}$ and $\mathcal{L}_T$ if XR was not given to it in the challenge step. $\mathcal{A}$ is said to have won the game $\mathcal{G}$ if:

1. $\mathsf{F}_{1,i}(x, r, \{u\}) = \mathsf{F}_{1,j}(x, r, \{u\})$ in $\mathbb{Z}_q$, for some two polynomials $\mathsf{F}_{1,i} \neq \mathsf{F}_{1,j}$ in $\mathcal{L}$.
2. $\mathsf{F}_{T,i}(x, r, \{u\}, \{v\}) = \mathsf{F}_{T,j}(x, r, \{u\}, \{v\})$ in $\mathbb{Z}_q$, for some two polynomials $\mathsf{F}_{T,i} \neq \mathsf{F}_{T,j}$ in $\mathcal{L}_T$.
3. $b' = b$.

This completes the description of the game $\mathcal{G}$ and simulator $\mathcal{C}$.

Let $\mathsf{Collision}$ denote either of the events 1 and 2 above, i.e. a *collision* occurring in lists $\mathcal{L}$ and/or $\mathcal{L}_T$. Denote the event 3 above by $\mathsf{Success}$.

Analysis of $\mathbf{Pr}^{\mathbf{CCA1}}_{\mathcal{A},\mathsf{BEG}}$**:** The success probability $\Pr^{\mathrm{CCA1}}_{\mathcal{A},\mathsf{BEG}}$ of $\mathcal{A}$ in the actual CCA1 game satisfies

$$\Pr^{\mathrm{CCA1}}_{\mathcal{A},\mathsf{BEG}} \leq \Pr[\mathsf{Success} \mid \overline{\mathsf{Collision}}] + \Pr[\mathsf{Collision}]. \tag{2.14}$$

This is because the event $\overline{\mathsf{Collision}}$ ensures that $\mathcal{A}$ will get to see only distinct group elements in the actual interaction. In other words, $\mathcal{A}$ is unable to cause *collisions* among group elements. As long as the event $\mathsf{Collision}$ does not occur,

then the view of $\mathcal{A}$ is identical in the game $\mathcal{G}$ and the actual interaction. Hence if $\mathcal{A}$ is unable to provoke collisions, then adaptive strategies are no more powerful than non-adaptive ones (see [Mau05, Lemma 2 on pp. 12], also [Sho97]). This observation allows us to choose group elements and their representations independently of the strategy of $\mathcal{A}$.

First we bound $\Pr[\mathsf{Collision}]$. The τ_1 polynomials $\mathsf{F}_{1,i}$ in $\mathcal{L}$ have degree at most one. Note that $\mathsf{F}_{1,i} \neq \mathsf{F}_{1,j} \Leftrightarrow \mathsf{F}_{1,i} - \mathsf{F}_{1,j} \neq 0$ as polynomials. From Lemma 2.3 (with $\lambda' = 0$), the probability that two distinct polynomials in $\mathcal{L}$ evaluate to the same value for randomly and independently chosen values for the indeterminates is at most $\frac{1}{q}$. Summing up over at most $\binom{\tau_1}{2}$ distinct pairs (i, j), the probability that the condition 1 above holds is at most $\binom{\tau_1}{2} \cdot \frac{1}{q}$. Similarly, the probability that the condition 2 above holds is at most $\binom{\tau_T}{2} \cdot \frac{2}{q}$. Since $\mathcal{A}$ makes at most $s_{\mathcal{O}} < s$ group oracle queries and that in each query $\mathcal{A}$ can guess at most two new elements, it is easy to see that lists $\mathcal{L}$ and $\mathcal{L}_T$ together have at most $3(s_{\mathcal{O}} + s_D) \leq 3s$ elements. Hence we obtain

$$\Pr[\mathsf{Collision}] \leq \binom{\tau_1}{2} \cdot \frac{1}{q} + \binom{\tau_T}{2} \cdot \frac{2}{q} \leq \frac{1}{q}(\tau_1 + \tau_T)^2 \leq \frac{9s^2}{q}. \tag{2.15}$$

Next, to bound $\Pr[\mathsf{Success} \mid \overline{\mathsf{Collision}}]$, we note that the adversary $\mathcal{A}$ will not be able to compute the polynomials XR or R from polynomials in $\mathcal{L}$ and $\mathcal{L}_T$ if XR or R was not given to it in the challenge step. Hence the event of no collision ensures that $\mathcal{A}$ will not be able to compute the representation of the element corresponding to XR or R. Hence $\Pr[\mathsf{Success} \mid \overline{\mathsf{Collision}}] = \frac{1}{2}$. Therefore, from (2.15) and (2.14), we get

$$\Pr^{\mathrm{CCA1}}_{\mathcal{A},\mathsf{BEG}} \leq \frac{1}{2} + \frac{9s^2}{q}. \tag{2.16}$$

Hence if $s = \mathrm{poly}(\log q)$, then $\Pr^{\mathrm{CCA1}}_{\mathcal{A},\mathsf{BEG}}$ is negligible. This completes the proof of Theorem 2.3. □

2.2.3.2 Leakage Setting: Completing Proof of Theorem 2.2.

Let us first briefly sketch the main ideas of the proof. Working on the lines of (2.14), the advantage of $\mathcal{A}$ is bounded by its success probabilities conditioned on the event whether or not a collision has occurred in the lists consisting of elements of $\mathbb{G}$ and $\mathbb{G}_T$. It is important to note that the proof for the non-leakage setting (i.e. proof of Theorem 2.3) and the leakage setting would be the same conditioned on the fact that a collision has not occurred, and that the leakage functions will not be able to compute the "polynomial X" corresponding to the secret key nor guess the correct representations of the group elements for which it only partially obtains information through the leakage functions. The reason is that in the event of no collision, the adversary gets to see only distinct group elements and hence it will not have enough

information on the relation amongst the group elements it can compute. The fact that the leakage functions cannot compute the full secret key shows that the adversary will never be able to continually leak the whole of the secret key. Hence leakage on the secret state will not be useful in this case. Hence the success probability of $\mathcal{A}$ is the same in the event of no collision (that includes the event of guessing the representations of group elements using partial information about them).

However the probability that a collision occurs in the leakage setting is increased by a factor of at most $2^{2\lambda}$. This is because when $\mathcal{A}$ has access to leakage output $f_i(\sigma_{i-1}, t_i)$ and $h_i(\sigma'_{i-1}, (t_i, Y_i))$ during i^{th} decryption query, then in adversary's view the parameters t_i $(i \geq 1)$ are no longer uniformly distributed even though they are still independent. Hence $\mathcal{A}$ can now cause collisions among polynomials (in Conditions 1-2 on page 37) with increased probability. Since t_i appears in both $f_i()$ and $h_i()$, its (average conditional) min-entropy will be reduced by at most 2λ bits.

The only useful information that the leakage functions can provide to $\mathcal{A}$ is about the secret key X. This is because the values t_i are independent of the derived shared secret key. However $\mathcal{A}$ can use the leakages of t_i to eventually leak X. If $\mathcal{A}$ is able to compute X, then it can trivially compute the symmetric key corresponding to the challenge ciphertext. The event of no collision, and the fact that X is not a "linear combination" of the inputs to the leakage functions, guarantees that $\mathcal{A}$ is unable to compute X. Note that because the representations of group elements in the GBG model are randomized, the probability of guessing the complete representations of each of σ_{i-1}, σ'_{i-1} and Y_i, given the leakages, is increased by a factor of at most $2^{2\lambda}$.

Proof. Let $\mathcal{A}$ be an s-query adversary that can break the security of the scheme BEG. Hence $\mathcal{A}$ can make totally at most s group oracle and pairing oracle queries ($s_{\mathcal{O}}$) and decryption oracle queries (s_D). In the count of s, even group oracle queries by leakage functions f_i, h_i $(i \geq 1)$ specified by $\mathcal{A}$ are also included. Let the adversary $\mathcal{A}$ play the game $\mathcal{G}'$ described below. This game is an extension of game $\mathcal{G}$ described in the proof of Theorem 2.3. To avoid repetition, we only describe here the extensions that are not part of game $\mathcal{G}$. Let $\{\mathsf{T}\}$ denote the list of indeterminates $\{\mathsf{T}_i : 1 \leq i \leq s_D\}$ that correspond to the values t_i in BEG.

Game $\mathcal{G}'$: For each leakage function $f_i(\sigma_{i-1}, t_i)$ and $h_i(\sigma'_{i-1}, (t_i, Y_i))$, $\mathcal{A}$ maintains a pair of lists $\left(\mathcal{L}^{f_i}, \mathcal{L}^{f_i}_T\right)$ and $\left(\mathcal{L}^{h_i}, \mathcal{L}^{h_i}_T\right)$, respectively. These lists contain polynomial and bit-string pairs. The polynomials in $\mathcal{L}^{f_i}$ and $\mathcal{L}^{h_i}$ belong to $\mathbb{Z}_q[\mathsf{X}, \mathsf{R}, \{\mathsf{U}\}, \{\mathsf{T}\}]$, and the corresponding bit-strings are from the encoding set Ξ of group $\mathbb{G}$. The polynomials in $\mathcal{L}^{f_i}_T$ and $\mathcal{L}^{h_i}_T$ are in the ring $\mathbb{Z}_q[\mathsf{X}, \mathsf{R}, \{\mathsf{U}\}, \{\mathsf{V}\}, \{\mathsf{T}\}]$, and the corresponding bit-strings are from the encoding set Ξ_T of group $\mathbb{G}_T$. Intuitively, the polynomials in lists $\mathcal{L}^{f_i}$ and $\mathcal{L}^{h_i}$ correspond to the elements of group $\mathbb{G}$ that can be computed by f_i and h_i,

respectively, whereas the lists $\mathcal{L}_T^{f_i}$ and $\mathcal{L}_T^{h_i}$ correspond to the elements of $\mathbb{G}_T$. Every polynomial in $\mathcal{L}^{f_i}$ is of the form $c_{1,i}\mathsf{T}_i + c_{2,i}\sum_{j=0}^{i-1}\mathsf{T}_j + c_{3,i}\mathsf{D}_i$, where $c_{1,i}$, $c_{2,i}$, $c_{3,i} \in \mathbb{Z}_q$ are chosen by $\mathcal{A}$ and $\mathsf{D}_i \in \mathbb{Z}_q[\mathsf{X}, \mathsf{R}, \{\mathsf{U}\}]$ is in $\mathcal{L}$ (cf. (2.12)). Every polynomial in $\mathcal{L}^{h_i}$ is of the form

$$d_{1,i}\mathsf{T}_i + d_{2,i}\left(\mathsf{X} - \sum_{j=0}^{i-1}\mathsf{T}_j\right) + d_{4,i}\mathsf{W}_i, \tag{2.17}$$

where $d_{1,i}$, $d_{2,i}$, $d_{3,i}$, $d_{4,i} \in \mathbb{Z}_q$ are also chosen by $\mathcal{A}$ and $\mathsf{W}_i \in \mathbb{Z}_q[\mathsf{X}, \mathsf{R}, \{\mathsf{U}\}]$ is in the list $\mathcal{L}$. Note that the polynomials in lists $\mathcal{L}^{f_i}$ and $\mathcal{L}^{h_i}$ are of degree at most one, and that they do not contain the monomial X. The polynomials in lists $\mathcal{L}_T^{f_i}$ and $\mathcal{L}_T^{h_i}$ are of degree at most two.

The game $\mathcal{G}'$ proceeds exactly as game $\mathcal{G}$ except that $\mathcal{A}$ can also obtain leakage through functions f_i and h_i in the i^{th} decryption query. The leakages on the representations of the group elements are simulated in the naive way, whereas for the leakages on t_i, a temporary random value is chosen for each t_i and the leakage on this value is given to the adversary. When $\mathcal{A}$ terminates it outputs a guess b' of b. Let us denote by $\mathsf{Success}^*$ the event of successful guess of the bit b by $\mathcal{A}$. Let $\mathsf{Collision}^*$ denote the event of a collision occurring in lists $\mathcal{L}$, $\mathcal{L}_T$, $\mathcal{L}^{f_i}$, $\mathcal{L}^{h_i}$, $\mathcal{L}_T^{f_i}$, $\mathcal{L}_T^{h_i}$ $(1 \le i \le s_D)$ and also the event of successful guessing of the partially leaked representations. The polynomials are now evaluated with values chosen from independent distributions with min-entropy $\log q - 2\lambda$, not necessarily from an uniform distribution. The exact distribution depends on the leakage functions chosen by $\mathcal{A}$. Since we are only interested to upper bound the collision probability, we can safely assume that the simulator chooses the right distribution. Note that even in the leakage setting, adaptive strategies are no more powerful than non-adaptive ones, as observed in [AM11, pp. 691]. This completes the description of the game $\mathcal{G}'$.

Let $\Pr^{\text{CCmLA1}}_{\mathcal{A},\mathsf{BEG}}$ denote the probability of the event $\mathsf{Success}^*$. On the lines of (2.14), we can write

$$\Pr^{\text{CCmLA1}}_{\mathcal{A},\mathsf{BEG}} \;\le\; \Pr[\mathsf{Success}^* \mid \overline{\mathsf{Collision}^*}] + \Pr[\mathsf{Collision}^*]. \tag{2.18}$$

As mentioned before, conditioned on the event $\overline{\mathsf{Collision}^*}$, the view of the adversary $\mathcal{A}$ will be same in both the games $\mathcal{G}'$ and $\mathcal{G}$. This is because in both the cases $\mathcal{A}$ will get to see only distinct group elements. Also, we are conditioning on the event that $\mathcal{A}$ will not be able to guess the correct representations of any of the at most $2\lambda s$ group elements it obtains through the leakage functions. Hence, on the lines of (2.16), we have

$$\Pr[\mathsf{Success}^* \mid \overline{\mathsf{Collision}^*}] \;=\; \frac{1}{2}. \tag{2.19}$$

Lemma 2.4. $\Pr[\mathsf{Collision}^*] \le \left(\frac{9s^2+2\lambda s}{q}\right) 2^{2\lambda}$.

Proof. To compute the required probability, the polynomials in lists $\mathcal{L}$, $\mathcal{L}_T$, $\mathcal{L}^{f_i}$, $\mathcal{L}^{h_i}$, $\mathcal{L}_T^{f_i}$, $\mathcal{L}_T^{h_i}$ $(1 \leq i \leq s_D)$ are evaluated by choosing values from $\mathbb{Z}_q$ according to (independent) distributions with min-entropy at least $\log q - 2\lambda$. This is because $\mathcal{A}$ can obtain at most 2λ bits of leakage about t_i $(i = 1, \ldots, s_D)$. According to Lemma 2.1, the values t_i have min-entropy at least $\log q - 2\lambda$ in the view of $\mathcal{A}$. The total length of all the lists is at most $3(\tau_1 + \tau_T) \leq 3s$) (c.f. 2.2.3.1). Working exactly on the lines of (2.15), and using Lemma 2.3 (with $\lambda' = 2\lambda$), we obtain this probability as $\frac{9s^2}{q}2^{2\lambda}$. The probability of the event that $\mathcal{A}$ will guess the complete representations of any of the at most $3s$ group elements, for which it can possibly obtain partial information on their representations through the leakage functions, is at most $\frac{3s \cdot 2^{2\lambda}}{q}$. Hence $\Pr[\mathsf{Collision}^*] \leq \left(\frac{9s^2+3s}{q}\right) 2^{2\lambda}$. □

From (2.18), (2.19) and Lemma 2.4, we have $\Pr^{\mathrm{CCmLA1}}_{\mathcal{A},\mathsf{BEG}} \leq \frac{1}{2} + \left(\frac{9s^2+3s}{q}\right) 2^{2\lambda}$. This completes the proof of Theorem 2.2. □

2.2.4 BEG-KEM+ : A Leakage-Resilient KEM Closer to Practice

Our choice of BEG-KEM for this investigation is entirely motivated by the fact that a similar leakage-resilience result as that proven in [KP10a] cannot be expected for a pairing-less group, as shown in Section 2.1. This motivates using pairing groups to implement ElGamal.

On the other hand, while Theorem 2.2 ensures a protection against side-channel attacks that combine traces of different computations (e.g. differential power analysis attacks), we still need protection against single trace attacks, i.e. Simple Power Analysis (SPA). The use of pairing groups can help on this respect, as pointed out by Scott in [Sco05]:

> "[...] it is of interest to consider the resistance of pairing-based protocols to so-called SPA attacks [...] one might with reasonable confidence expect that the power consumption profile of (and execution time for) such protocols will be constant and independent of any secret values."

We continue by proposing a tweak to BEG-KEM with the aim to make the most, from a minimizing leakage perspective, out of our choice of using pairing groups to realize leakage-resilient public-key cryptographic primitives.

2.2.4.1 An Advanced BEG-KEM+

Let us first make the observation that $\mathsf{Dec1}^*_{\mathsf{BEG}}$ is picking a random point in the pairing based group $\mathbb{G}$ by computing an exponentiation g^r for a random

r. As is well-known, a naïve implementation of exponentiation can leak the entire exponent r, which would, of course, invalidate the required bound on maximum leakage in our new (as well as in the old) model. This leads us to the question whether it is possible, given the large body of side-channel resistant exponentiation techniques, to find an algorithm that would likely meet the leakage bound for single measurements. In other words, we have to answer the question of whether the exponentiation can be made resistant against SPA attacks.

Exponentiation in a multiplicative group (or scalar multiplication in an elliptic curve group) of large order involves hundreds or even thousands of low-level arithmetic operations such as modular multiplication. Unfortunately, all these low-level operations are (either directly or indirectly) controlled by the secret exponent, which means that each of them can potentially leak sensitive information (see e.g. [WT01, Wal04, ST06] for further details). Consequently, we need both an SPA-resistant exponentiation algorithm and an SPA-resistant implementation of the underlying multiple-precision operations. The latter is difficult to achieve in software due to side-channel leakage induced by certain micro-architectural features such as the early-termination mechanism of integer multipliers in ARM processors [GOPT10]. For example, it was shown in [GOPT10] that highly regular exponentiation (resp. scalar multiplication) techniques, which are (in theory) perfectly SPA-resistant, succumb to an SPA attack when exploiting the early-termination mechanism. Therefore, we avoid exponentiation with a secret exponent in our modified scheme[2].

A careful analysis of BEG-KEM reveals that $\mathsf{Dec1}^*_{\mathsf{BEG}}$ only needs to sample uniformly at random an element u of $\mathbb{G}$, and that knowledge of $\log_g u$ is *not necessary*. For this reason, we decided to build a random u in the pairing base group by using a so-called *encoding* to the base group [SvdW06, Ica09, FT12]. Roughly speaking, an encoding is a deterministic function mapping an arbitrary string to a point in an elliptic curve. Recently, Fouque and Tibouchi [FT12] proposed a modification of the Shallue and van de Woestijne encoding into arbitrary elliptic curves [SvdW06], that maps arbitrary strings to Barreto-Naehrig asymmetric pairing groups [BN05]. Let $f : \mathbb{F}_p^* \to E(\mathbb{F}_p)$ be the Fouque-Tibouchi encoding. Then, $(t_1, t_2) \mapsto u = u_1 \cdot_E u_2$ builds a point $u \in E(\mathbb{F}_p)$ distributed uniformly at random if $t_1, t_2 \xleftarrow{\$} \mathbb{F}_p^*$, where $\cdot_E$ is the addition operation in $E(\mathbb{F}_p)$. Additionally, [FT12] points out that f can be naturally implemented so that its computation is completely independent of the inputs, which clearly helps us towards meeting our desired min-entropy leakage bound.

[2] As mentioned previously, the secret exponent controls a large number of multiple-precision arithmetic operations, which execute an even larger number of `mul` instructions. Each of these `mul` instructions can potentially trigger the early-termination mechanism and, hence, leak information about the secret exponent. In our modified scheme, the secret value is only used as input of a multiple-precision operation and does not control any other operations.

Algorithm 1 Shallue-van de Woestijne encoding to BN curves $y^2 = x^3 + b$ [FT12]

Input: A random number $t \in \mathbb{F}_p^\star$.
Output: Point $P \in E(\mathbb{F}_p)$

1: $w \leftarrow \sqrt{-3} \cdot t/(1 + b + t^2)$
2: $x_1 \leftarrow (-1 + \sqrt{-3})/2 - tw$
3: $x_2 \leftarrow -1 - x_1$
4: $x_3 \leftarrow 1 + 1/w^2$
5: $r_1, r_2, r_3 \xleftarrow{\$} \mathbb{F}_p^\star$
6: $\alpha \leftarrow \chi_q(r_1^2 \cdot (x_1^3 + b))$
7: $\beta \leftarrow \chi_q(r_2^2 \cdot (x_2^3 + b))$
8: $i \leftarrow [(\alpha - 1) \cdot \beta \bmod 3] + 1$
9: **return** $P[x_i, \chi_q(r_3^2 \cdot t) \cdot \sqrt{(x_i^3 + b)}]$

2.2.4.2 BEG-KEM+: Description

Let ABGen be an asymmetric bilinear group generator that outputs $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, q, g_1, g_2)$ with $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| = q$, where q is a prime, κ be the security parameter, and λ be the leakage parameter. We will again use the multiplicative notation for group operations in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a type 3 pairing map, i.e., e is a non-degenerate bilinear map with no known efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$. These groups are instantiated using the BN curves, denoted $E(\mathbb{F}_p)$, of the form $y^2 = x^3 + b$, where $b \in \mathbb{F}_p$ [BN05]. Also, let G_1 and G_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $f : \mathbb{F}_p^* \to \mathbb{G}_1$ be the Fouque-Tibouchi encoding of the elements of $\mathbb{G}_1$.

The advanced $\mathsf{BEG-KEM+} = \left(\mathsf{KG}^+_{\mathsf{BEG}}, \mathsf{Enc}^+_{\mathsf{BEG}}, \mathsf{Dec1}^+_{\mathsf{BEG}}, \mathsf{Dec2}^+_{\mathsf{BEG}}\right)$ is defined as follows:

1. $\mathsf{KG}^+_{\mathsf{BEG}}(\kappa)$: Compute $\mathbb{PP} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, q, G_1, G_2) \leftarrow \mathsf{ABGen}(\kappa)$ and randomly choose $x, t_0 \xleftarrow{\$} \mathbb{F}_q$. Set $X = G_1^x$, $\sigma_0 = G_1^{t_0}$, $\sigma_0' = G_1^{(x-t_0)}$, and $X_T = e\,(G_1, G_2)^x$. Return (pk, sk_0), where
   (a) the public key is $pk = (\mathbb{PP}, X_T)$.
   (b) the secret state is $sk_0 = (\sigma_0, \sigma_0')$.
2. $\mathsf{Enc}^+_{\mathsf{BEG}}(pk)$*:* Choose a random $r \xleftarrow{\$} \mathbb{F}_p$. Compute the ciphertext $C = G_2^r$, and the derived key $K = X_T^r$. Return (C, K).
3. $\mathsf{Dec1}^+_{\mathsf{BEG}}(\sigma_{i-1}, C)$: Choose random $t_i, z_i \xleftarrow{\$} \mathbb{F}_p^*$, set $u_i = f\,(t_i) \cdot f\,(z_i)$, and compute $\sigma_i = \sigma_{i-1} \cdot u_i$, $Y_i = e\,(\sigma_i, C)$. Return (u_i, Y_i).
4. $\mathsf{Dec2}^+_{\mathsf{BEG}}(\sigma_{i-1}', (u_i, Y_i)\,, C)$: Set $\sigma_i' = \sigma_{i-1}' \cdot (u_i)^{-1}$, and $Y_i' = e\,(\sigma_i', C)$. Compute the derived key $K = Y_i \cdot Y_i' \in \mathbb{G}_T$. Return K.

Algorithm 1 describes the constant-time hashing function to BN curves from [FT12]. As described in the original paper, implementing this algorithm

against timing and Simple Power Analysis (SPA) attacks is not difficult to be achieved. In step 6 and 7, instead of computing the values $\chi_q(x_1^3 + b)$ and $\chi_q(x_2^3 + b)$ in a straightforward way, which can leak secret data, the authors suggested to use blinding. Namely, in order to get α and β, we actually evaluate $\chi_q(r_1^2 \cdot (x_1^3 + b))$ and $\chi_q(r_2^2 \cdot (x_2^3 + b))$, where r_1 and r_2 are random field elements generated in Step 5. On the other hand, in order to prevent the leakage while computing the index i, they employ a specific algebraic function $\phi(\alpha, \beta) = [(\alpha - 1) \cdot \beta \bmod 3] + 1$, which runs in constant time.

2.2.5 Secure Implementation and Performance Analysis

In this section, we first describe a software implementation of BEG-KEM+ (along with the instantiation of the underlying pairing groups) and present the execution times we measured on an ARM Cortex M-3 processor. The second part of this section is devoted to a "practical" security evaluation of BEG-KEM+ by analyzing potential sources of information leakage in the underlying arithmetic operations that could be exploited to mount a side-channel attack.

2.2.5.1 Implementation Details and Performance Analysis

We implemented both BEG-KEM and BEG-KEM+ in Magma and ANSI C, whereby the former implementation served as a reference for the latter. The C implementation is based on the MIRACL library to ensure an efficient execution of the pairing evaluation and all other arithmetic operations in the diverse groups and fields. We instantiated both BEG-KEM and our improved scheme using the Ate pairing over a 254-bit Barreto-Naehrig (BN) curve. More specifically, our implementations adopts the curve BN254 from [PSNB11], which provides a security level roughly comparable to that of 128-bit AES. BN curves are defined by a Weierstrass equation of the form $y^2 = x^3 + b$ over a prime field $\mathbb{F}_q$, whereby q can be written as polynomial $p(u) = 36u^4 + 36u^3 + 24u^2 + 6u + 1$ for some parameter u [BN05]. In our case, $u = -(2^{62} + 2^{55} + 1) = -0\text{x}4080000000000001$ and, hence, q has a length of 254 bits. The curve BN254 is given by the equation $y^2 = x^3 + 2$ (i.e. $b = 2$) and has prime order with embedding degree $k = 12$.

The execution times for various arithmetic operations in the different fields and groups are summarized in Table 2.2, whereby all timings are specified in millions of clock cycles. Our prototype platform for performance evaluation is an Arduino Due microcontroller board equipped with an ARM Cortex-M3 CPU. Even though the three groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ have the same order, the underlying multiple-precision arithmetic operations are performed with operands of different size. $\mathbb{G}_1$ and $\mathbb{G}_2$ are elliptic curve groups over $\mathbb{F}_q$ and $\mathbb{F}_{q^2}$, the elements of which have, in our case, a bitlength of 254 and 508 bits, respectively. The group $\mathbb{G}_T$ is a subgroup of the multiplicative group of the

extension field $\mathbb{F}_{q^{12}}$, i.e. the modular multiplications for exponentiation in $\mathbb{G}_T$ are carried out on 3048-bit operands.

Table 2.2: Running times for field exponentiation, square root, inversion, group exponentiation and pairing operations (in 10^6 clock cycles)

Operation	Running time
Square root $\mathbb{F}_q$	0.7
Inversion $\mathbb{F}_q$	0.087
Encoding to $\mathbb{G}_2$	3.7
Exponentiation $\mathbb{G}_1$	4.5
Exponentiation $\mathbb{G}_2$	10.0
Exponentiation $\mathbb{G}_T$	27.1
Pairing	65.0

Table 2.3: Comparison of running times for key generation, encapsulation and decapsulation for BEG-KEM and BEG-KEM+ (in 10^6 clock cycles)

Operation	BEG-KEM	BEG-KEM+
KeyGen	108	108
Encryption	34	34
Decryption	131	140

The execution times for key generation, encapsulation as well as decapsulation for both BEG-KEM and BEG-KEM+ are given in Table 2.3. Our results show that an encapsulation can be carried out in 34 million clock cycles, while the decapsulation takes about 140 million cycles. We observe that our modified decapsulation algorithm is roughly 6% slower than the original one.

2.2.5.2 Side-Channel Resistance: Practical Point of View

One of the fundamental principles of leakage-resilient cryptography is to use a critical secret only once (or a few times), which ensures that an attacker is not able to retrieve the secret key if the per-invocation leakage is in some way "limited" or "bounded." In every invocation of the scheme or function, the secret is either "refreshed" or a completely new secret is generated randomly. The original BEG-KEM scheme from [KP10a], and also our variant BEG-KEM+, follow this principle. As a consequence, all forms of side-channel attack that require several executions of a cryptographic function with one and the same secret key, e.g. Differential Power Analysis (DPA), are obviously not applicable to BEG-KEM+ (and in fact the latter is guaranteed by Theorem 2.2). However, attacks that aim to recover the secret key from information leaked from a single invocation of a cryptographic function (i.e. Simple Power Analysis (SPA) attacks) may succeed under certain conditions. The group exponentiation computed in the BEG-KEM scheme to derive a random group element $\sigma_0 = g^{t_0}$ serves as a good example. If this exponentiation is implemented in completely straightforward way (e.g. using the square-and-multiply method) an attacker can obtain t_0 if he is able to distinguish group squarings from group products in the power consumption profile. Such SPA attacks on unprotected or insufficiently protected ECC implementations are fairly easy and have been reported extensively in the literature, see e.g. [BSS05, Chapter IV] and the references therein. Therefore, we advocated to replace the

afore-mentioned group exponentiation by a deterministic encoding into an elliptic curve group [FT12].

SPA Resistance of Pairing Evaluation. Section 2.2.4.1 quotes a statement of Scott [Sco05, Section 3.1] saying that one can expect the power consumption profile of a pairing-based protocol to be independent of any secret values. An intuitive explanation why pairings are fairly "robust" against SPA leakage is also given in [Sco05]: the target of the attack is a secret point, which is generally much harder to reveal than e.g. a secret scalar or a secret exponent. As mentioned before, our implementation uses the Ate pairing instantiated on a BN curve over a 254-bit prime field $\mathbb{F}_p$. Consequently, the secret is the x and y coordinate of an elliptic curve point, which are in our case simply elements of $\mathbb{F}_p$. The only way in which an attacker can hope to gain information about x and y is by inspecting the power consumption and execution time of the $\mathbb{F}_p$-arithmetic operations (e.g. addition, multiplication) performed on them. However, the operand-related SPA leakage from field-arithmetic operations is generally very small. To explain this in detail, let us use the addition in $\mathbb{F}_p$ as example, which is nothing else than a modular addition $r = a + b \bmod p$. We assume that a is a secret value and that b is known to the attacker. A modular addition consists of an ordinary addition $s = a + b$, followed by a subtraction if the sum s is equal to or bigger than p. Conventional wisdom from the side-channel community says that such a conditional subtraction causes differences in the power consumption profile (and also execution time), which is observable by an attacker. However, the information content is very small; in fact, when the subtraction is executed the attacker just knows that $a + b \geq p$, i.e. he has learned that $a \geq p - b$.

The situation is similar for multiplication in $\mathbb{F}_p$, which is nothing else than a modular multiplication $r = a \cdot b \bmod p$. Again, we assume that a is the secret value and that b is known to the attacker. A modular multiplication involves a conventional multiplication $t = a \cdot b$, followed by a modular reduction $r = t \bmod p$, which is in pairing-based cryptography typically implemented using Montgomery's algorithm [Mon85]. Both the multiplication and Montgomery reduction are highly regular (i.e. do not have to execute any conditional statements), except for the so-called "final subtraction." Montgomery's reduction technique does not directly compute $t \bmod p$ but produces the following output

$$x = \big(t + (t \cdot p' \bmod 2^n) \cdot p\big)/2^n \tag{2.20}$$

where $p' = -p^{-1} \bmod 2^n$ and n is the bitlength of p. Note that x may be not fully reduced, which means a final subtraction of p is necessary to get the least non-negative residue as result. An attacker able to observe whether or not this final subtraction is executed learns only whether $x \geq p$ or not, which does not reveal much information about a. The same also holds for subtraction and squaring in $\mathbb{F}_p$. However, a noteworthy exception is the inversion operation, which we will further discuss below. In summary, a straightforward

implementation of the arithmetic operations (bar inversion) in $\mathbb{F}_p$ leaks only very little information about the operands, which confirms that pairing evaluation is, in general, not susceptible to SPA attacks. To our knowledge, the recent literature contains only two papers in which SPA attacks on pairings are discussed [PV04, WS06], but both of them are only relevant for pairings over binary fields where the multiplication is implemented in a highly irregular way. The attack from [Wal04] is only applicable to scalar multiplication with a secret scalar, but not to pairings with a secret point.

SPA Resistance of Encoding Function. The encoding function shown in Algorithm 1 consists of a number of basic arithmetic operations (e.g. addition, multiplication) in the field $\mathbb{F}_p$. Furthermore, two inversions are executed, one in step 1 and the other in step 4. The straightforward approach to invert an element of a finite field is the Extended Euclidean Algorithm (EEA). Conventional wisdom from the side-channel community says that the EEA is a highly irregular algorithm, executing many conditional operations, which is likely to leak SPA-relevant information about the operand to be inverted. In order to prevent an SPA attack on the inversion operation, we apply a simple multiplicative masking. That is, instead of inverting a field element v directly, we first multiply it by a random number r, which yields the product $t = v \cdot r$. Then, we invert this product using the EEA to obtain $1/t = 1/(v \cdot r)$, which we finally multiply again by r to get $1/v$ as result.

The function χ in step 6 and 7 of Algorithm 1 is essentially an evaluation of the Legendre Symbol, which, in turn, consists of an exponentiation using a constant public exponent (i.e. $(p+1)/4$). The input to the χ function is "blinded" by the random value r_1^2 and r_2^2, which means the underlying exponentiation can not leak any SPA-relevant information. As mentioned in Section 2.2.4.1, a constant-time algebraic function is adopted for the calculation of the index i in step 8, which also cannot leak.

2.3 Conclusion and Future Directions

In the first part of this chapter, we cryptanalyzed a variant of the ElGamal key encapsulation mechanism proposed in [KP10a]. Though the attack model of [KP10a] allows an attacker to query with (bounded) leakage functions in the split-state model, we really need simple leakage functions for our attack - leakage of $\left(\frac{3}{8} + o\left(1\right)\right)$ fraction of the most significant bits of the current secret state and the past secret state, alternatively, in successive rounds. We do not need an adversary to have access to decryption oracle, and our attack works for any arbitrary group used to instantiate the scheme.

For future work related to this part, it would be interesting to see whether Coppersmith-like lattice techniques [Cop97] could lead to attacks that require lesser fraction of the most significant bits of the secret shares. Another direc-

tion is to rigourously prove our leakage bound of $\left(\frac{3}{8} + o\left(1\right)\right)$, possibly using the techniques of Ling et al. [LSSW12]. A challenging open problem is to address the hidden shares - hidden number problem over pairing groups.

In the latter part of this chapter, first, we argued that a naive implementation of the pairing group exponentiation in the leakage-resilient ElGamal key encapsulation mechanism proposed in [KP10a] makes it impossible to reach the required leakage bound. To overcome this problem, we have made two additional contributions. On the one hand, we have proposed a relaxed leakage model, that we call min-entropy leakage, that lifts the restriction on the image size of leakage functions, and proposes instead to require that the inputs to the leakage functions have sufficient min-entropy left, in spite of the leakage. On the other hand, we adopted a different mechanism for finding a random point in an elliptic curve group, namely the encoding of Fouque and Tibouchi. We assessed the security of our implementation from both a theoretical and a practical perspective and argued that it is indeed secure in both the worlds. BEG-KEM+ is, to our knowledge, the first leakage-resilient public-key scheme that has been successfully implemented and evaluated on an embedded 32-bit processor.

As far as we know, no stateful ElGamal-based public-key encryption scheme with constant public-key size exists in the literature offering provable resistance against a broad class of continual leakage attacks in the standard model. Finding such a scheme remains a challenging open question.

Another interesting direction is to obtain encryption schemes of efficiency comparable to that of BEG-KEM but secure against a stronger leakage model than the split-state model, possibly still using the generic group model to argue about the security.

Chapter 3

Split-State Boneh–Boyen Signature Scheme

In this chapter, we propose a leakage-resilient signature scheme in the continual split-state leakage model that is based on a well-known identity-based encryption scheme by Boneh and Boyen [BB04]. The proposed signature scheme is the one of the most efficient among the existing schemes that allow for continual leakage. Its efficiency is close to that of non leakage-resilient pairing-based signature schemes. It tolerates, asymptotically, leakage of half of the bits of the secret key at every new signature invocation. We prove the security of the new scheme in the generic bilinear group model. We observe that the original Boneh–Boyen signature scheme is existentially unforgeable in the generic bilinear group model, whereas it is only known to be selectively unforgeable in the standard model.

Contents

3.1 Introduction

In Chapter 2, we analyzed a pairing-based variant of the ElGamal encryption scheme due to Kiltz and Pietrzak [KP10a] in the min-entropy leakage model. In this chapter, we use the techniques from Chapter 2 to propose a leakage-resilient signature scheme that builds upon the Boneh–Boyen identity-based encryption scheme [BB04].

We make two main assumptions to model leakage:

- **bounded leakage:** the useful leakage data per signature invocation is bounded in length, but unbounded overall;
- **split-state leakage:** the computation can be divided into rounds, where each such round leaks independently.

The first assumption, probably unlike the min-entropy model, can be seen overly restrictive; however it should be noticed that in practice many side-channel attacks only exploit a polylogarithmic amount of information. Note that the second assumption is also present in the min-entropy leakage model.

Our variant of the Boneh–Boyen signature scheme is obtained by splitting the secret group element into two parts that is continually refreshed. The resulting signature scheme is nearly as efficient as the original identity-based encryption scheme, being only $\frac{4}{3}$ times slower. Our security reduction is in the generic bilinear group (GBG) model. Our main theorem (Theorem 3.2) states that allowing λ bits of leakage at every round decreases the security of the scheme by at most a factor $2^{2\lambda}$. A interesting observation from our analysis is that the Boneh–Boyen signature scheme is existentially unforgeable in the GBG model, whereas it is only known to be selectively unforgeable in the standard model.

Finally, we note that our proposal is far more efficient than signature schemes proven to be leakage-resilient under similar (or more stronger) leakage model, with proof in the standard model [KV09, FKPR10, BKKV10, DHLAW10b, MTVY11, BSW11].

We start in Section 3.2 by recalling some basic facts and definitions. In Section 3.3, we prove the security, without leakage, of the Boneh–Boyen signature scheme in the GBG model. In Section 3.4, we split the secret state of the scheme and prove its leakage resilience under continual split-state leakage in the GBG model.

3.2 Definitions

In this section, we recollect some basic notions of security for signature schemes. We also describe the model of leakage we shall consider in this chapter and formulate a definition of security of signature schemes in the presence of continual leakage. We adapt the leakage model specified in [KP10a] to signature schemes.

For other necessary technical background for this chapter, refer to Section 2.2.2.1 for a discussion on min-entropy, to Section 2.2.2.2 on bilinear groups, and to Section 2.2.2.3 on the generic bilinear group model.

A signature scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ consists of three probabilistic polynomial-time algorithms KeyGen, Sign, and Verify. Let κ denote the security parameter. $\mathsf{KeyGen}(\kappa)$ on input κ produces a public- and secret-key pair

(pk, sk) along with other public parameters $\mathbb{PP}$. The algorithm $\mathsf{Sign}(sk, m)$ on input a secret key sk and a message $m \in M$, where M is the message space, outputs a signature σ. $\mathsf{Verify}(pk, m, \sigma)$ on input a public key pk, a message $m \in M$ and a signature σ, outputs a bit $b = 1$ meaning *valid*, or $b = 0$ meaning *invalid*. We require the following *correctness* requirement to be satisfied by Π:

$$\Pr[\mathsf{Verify}(pk, m, \mathsf{Sign}(sk, m)) = 1 \ : \ (pk, sk) \leftarrow \mathsf{KeyGen}(\kappa), m \in M] = 1.$$

The standard security notion for signature schemes is existential unforgeability under adaptive chosen-message attacks (EUF-CMA), and it is is defined through the following experiment:

Sign-Forge$_{\Pi}(\mathcal{A}, \kappa)$	Sign-Oracle $\Omega_{sk}(m)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\kappa)$	$w := w \cup m$
$w := \emptyset$	$\sigma \leftarrow \mathsf{Sign}(sk, m)$
$(m, \sigma) \leftarrow \mathcal{A}^{\Omega_{sk}(\cdot)}(pk)$	Return σ
If $m \in w$, then return $b := 0$	
$b \leftarrow \mathsf{Verify}(pk, m, \sigma)$	

Definition 3.1. [Existential Unforgeability] *A signature scheme* Π *is* existentially unforgeable under adaptive chosen-message attacks*, in short "secure", if* $\Pr[b = 1]$ *is negligible in* Sign-Forge$_{\Pi}(\mathcal{A}, \kappa)$ *for any efficient adversary* $\mathcal{A}$.

3.2.1 Leakage Model

We split the secret state into two parts that reside in different parts of the memory, and structure any computation that involves access to the secret state into a sequence of steps. Any step accesses only one part of the secret state (*active* part) and the other part (*passive* part) is assumed not to leak in the current step of computation. For simplicity, we define a security notion for leakage-resilient signature schemes assuming that the signing process is carried out in two steps. We also refer to a single invocation of the signature generation algorithm as a *round*.

Let us consider the problem of achieving leakage resilience under continual leakage even when a significant fraction of the bits of the secret state are leaked per round. Then it is necessary that the secret state must be *stateful*, i.e. the secret state must be refreshed during every round [KP10a]. Otherwise, after many rounds the entire secret state will be completely leaked.

Formally, a stateful signature scheme $\Pi^* = (\mathsf{KeyGen}^*, \mathsf{Sign}_1^*, \mathsf{Sign}_2^*, \mathsf{Verify}^*)$ consists of four probabilistic polynomial-time algorithms KeyGen^*, Sign_1^*, Sign_2^* and Verify^*. $\mathsf{KeyGen}^*(\kappa)$ is same as the set-up phase KeyGen of Π except that instead of a "single" secret key sk, it outputs two initial secret states (S_0, S_0'). From the point of view of an adversary, the signing algorithm Sign of Π and

$(\mathsf{Sign}_1^*, \mathsf{Sign}_2^*)$ have the same functionality. First, Sign_1^* is executed and later Sign_2^* is executed. That is, the i^{th} round of the signing process is carried out as:

$$(S_i, w_i) \xleftarrow{r_i} \mathsf{Sign}_1^*(S_{i-1}, m_i) \; ; \; (S'_i, \sigma_i) \xleftarrow{r'_i} \mathsf{Sign}_2^*(S'_{i-1}, w_i). \tag{3.1}$$

In the above expression, r_i and r'_i are the randomness used by Sign_1^* and Sign_2^*, respectively. The parameter w_i is some state information passed onto Sign_2^* by Sign_1^*. The signature σ_i is generated for the message m_i, and the internal state is updated from (S_{i-1}, S'_{i-1}) to (S_i, S'_i).

We model the leakage during signature generation by giving an adversary $\mathcal{A}$ access to a leakage oracle $\Omega^{\text{leak}}_{(S_{i-1}, S'_{i-1})}(\cdot)$. This oracle, in addition to giving $\mathcal{A}$ signatures for the messages of its choice, also allows $\mathcal{A}$ to obtain leakage from the computation used to generate signatures. More precisely, let λ be a *leakage parameter*. During the i^{th} signing round, $\mathcal{A}$ is allowed to specify two functions f_i and h_i, each of range $\{0,1\}^\lambda$, that can be efficiently computed. The outputs of the leakage functions are

$$\Lambda_i = f_i(S_{i-1}, r_i) \; ; \; \Lambda'_i = h_i(S'_{i-1}, r'_i, w_i). \tag{3.2}$$

Since the value of m can be included in the description of f_i and h_i, hence it is not explicitly included as an input. Note that it also possible for $\mathcal{A}$ to specify h_i after obtaining Λ_i. But for simplicity of the exposition, we only describe here the case where f_i and h_i are specified along with the message m_i to the oracle. The security of the signature scheme Π^* in the presence of (continual) leakage is defined through the following experiment $\text{Sign-Forge-Leak}_{\Pi^*}(\mathcal{A}, \kappa, \lambda)$. In the description below, $|f_i|$ refers to the length of the output of f_i.

$\text{Sign-Forge-Leak}_{\Pi^*}(\mathcal{A}, \kappa, \lambda)$	Sign-Leak-Oracle $\Omega^{\text{leak}}_{(S_{i-1}, S'_{i-1})}(m_i, f_i, h_i)$
$(pk, (S_0, S'_0)) \leftarrow \mathsf{KeyGen}^*(\kappa)$	If $\lvert f_i\rvert \neq \lambda$ or $\lvert h_i\rvert \neq \lambda$, return $\perp$
$i := 1, w := \emptyset$	$(S_i, w_i) \xleftarrow{r_i} \mathsf{Sign}_1^*(S_{i-1}, m_i)$
$(m, \sigma) \leftarrow \mathcal{A}^{\Omega^{\text{leak}}_{(S_{i-1}, S'_{i-1})}(\cdot)}(pk)$	$(S'_i, \sigma_i) \xleftarrow{r'_i} \mathsf{Sign}_2^*(S'_{i-1}, w_i)$
If $m \in w$, then return $b := 0$	$\Lambda_i := f_i(S_{i-1}, r_i)$
$b \leftarrow \mathsf{Verify}^*(pk, m, \sigma)$	$\Lambda'_i := h_i(S'_{i-1}, r'_i, w_i)$
	$i := i + 1$
	$w := w \cup m_i$
	Return $(\sigma_i, \Lambda_i, \Lambda'_i)$

Definition 3.2. [Existential Unforgeability with Leakage] *A signature scheme* Π^* *is* existentially unforgeable under adaptive chosen-message attacks in the presence of (continual) leakage *if* $\Pr[b = 1]$ *is negligible in the Experiment* $\text{Sign-Forge-Leak}_{\Pi^*}(\mathcal{A}, \kappa, \lambda)$ *for any efficient adversary* $\mathcal{A}$.

3.3 Basic Boneh–Boyen Signature Scheme

We now describe a signature scheme that is obtained from the Boneh–Boyen identity based encryption scheme (BB-IBE) [BB04]. This scheme is not yet known to be existentially unforgeable under adaptive chosen-message attacks (EUF-CMA) in the standard model. However, we are able to prove that the BB-signature scheme is EUF-CMA secure in the GBG model.

Let $\Pi_{\mathsf{BB}} = (\mathsf{KeyGen}_{\mathsf{BB}}, \mathsf{Sign}_{\mathsf{BB}}, \mathsf{Verify}_{\mathsf{BB}})$ be a signature scheme on the message space $\mathbb{Z}_p$ defined as follows:

1. $\mathsf{KeyGen}_{\mathsf{BB}}(\kappa)$: Compute $\mathbb{PP} := (\mathbb{G}, \mathbb{G}_T, p, e, g) \leftarrow \mathsf{BGen}(\kappa)$. Choose random $x, x_0, x_1 \leftarrow \mathbb{Z}_p$. Set $X := g^x$, $X_0 := g^{x_0}$, $X_1 := g^{x_1}$ and $X_T := e(g, X) = e(g, g)^x$. The public key is $pk := (\mathbb{PP}, X_0, X_1, X_T)$ and the secret key is $sk := X$.
2. $\mathsf{Sign}_{\mathsf{BB}}(sk, m)$: Choose a random $t \leftarrow \mathbb{Z}_p$. Set $\sigma := (sk \cdot (X_0 \cdot X_1^m)^t, g^t)$. Output the signature σ.
3. $\mathsf{Verify}_{\mathsf{BB}}(pk, m, \sigma)$: Let $\sigma = (\sigma_1, \sigma_2) \in \mathbb{G}^2$. Output the bit $b = 1$ (*valid*) if $X_T \star e(\sigma_2, X_0 \cdot X_1^m) = e(\sigma_1, g)$. Otherwise output $b = 0$ (*invalid*).

Theorem 3.1. *The signature scheme* Π_{BB} *is* EUF-CMA *secure in the generic bilinear group model.*

Proof. Let $\mathcal{A}$ be a q-query adversary that can break the security of Π_{BB}. By a q-query adversary we mean that $\mathcal{A}$ can make totally at most q group oracle and signing oracle queries. Let $q_{\mathcal{O}}$ be the total number of calls to the group oracles $\mathcal{O}$, $\mathcal{O}_T$ and $\mathcal{O}_e$, and q_Ω correspond to the number of calls to the signing oracle. We have $q_{\mathcal{O}} + q_\Omega \leq q$.

As is typical for proofs in the generic group model, we bound the advantage of $\mathcal{A}$ against Π_{BB} by the success probability of $\mathcal{A}$ in the following game $\mathcal{G}$ simulated by the challenger $\mathcal{B}$ (see [Sho97, Mau05, BB08]). $\mathcal{B}$ simulates the EUF-CMA experiment in the naïve way. $\mathcal{B}$ answers the signature queries in a straightforward manner, and simulates the bilinear generic group oracles by maintaining a list of pairs (for each group) that represents the relation among the group elements output by group oracles or guessed by $\mathcal{A}$, and their corresponding encodings. We then argue that a forgery cannot be computed (as a "linear combination") from the group elements known to $\mathcal{A}$ (cf. Lemma 3.1), except with negligible probability.

Game $\mathcal{G}$: Let X, X_0, X_1, $\{\mathsf{T}_i : 1 \leq i \leq q_\Omega\}$, $\{\mathsf{U}_i : 1 \leq i \leq q_g,\ 0 \leq q_g \leq 2(q_{\mathcal{O}} + 1)\}$ and $\{\mathsf{V}_i : 1 \leq i \leq q_{g_T},\ 0 \leq q_{g_T} \leq 2q_{\mathcal{O}}\}$ be indeterminates, and $\{m_i : 1 \leq i \leq q_\Omega\}$ be elements of $\mathbb{Z}_p$ chosen by $\mathcal{A}$. Intuitively, these indeterminates correspond to randomly chosen group elements in Π_{BB}, or more precisely their discrete logarithms. The indeterminates X, X_0, X_1 correspond to the quantities x, x_0, x_1, respectively. Note that $\mathcal{A}$ might query the group oracles with representations (bit-strings) not previously obtained

from the group oracles. In order to accommodate this case we introduce the indeterminates U_i, V_i. The U_i $(1 \le i \le q_g)$ correspond to the elements of $\mathbb{G}$, whereas V_i $(1 \le i \le q_{g_T})$ correspond to the elements of $\mathbb{G}_T$. We denote the lists $\{\mathsf{T}_i : 1 \le i \le q_\Omega\}$, $\{\mathsf{U}_i : 1 \le i \le q_g\}$ and $\{\mathsf{V}_i : 1 \le i \le q_{g_T}\}$ by $\{\mathsf{T}\}$, $\{\mathsf{U}\}$ and $\{\mathsf{V}\}$, respectively.

$\mathcal{B}$ maintains two lists of pairs

$$\mathcal{L} = \{(\mathsf{F}_{1,i}\,,\, \xi_{1,i}) : 1 \le i \le \tau_1\}, \tag{3.3}$$

$$\mathcal{L}_T = \{(\mathsf{F}_{T,i}\,,\, \xi_{T,i}) : 1 \le i \le \tau_T\}, \tag{3.4}$$

such that, at step τ $(0 \le \tau \le q_{\mathcal{O}})$ in the game,

$$\tau_1 + \tau_T = \tau + 2q_\Omega + q_g + q_{g_T} + 4. \tag{3.5}$$

The entries $\mathsf{F}_{1,i} \in \mathbb{Z}_p[\mathsf{X}, \mathsf{X}_0, \mathsf{X}_1, \{\mathsf{U}\}, \{\mathsf{T}\}]$, $\mathsf{F}_{T,i} \in \mathbb{Z}_p[\mathsf{X}, \mathsf{X}_0, \mathsf{X}_1, \{\mathsf{U}\}, \{\mathsf{V}\}, \{\mathsf{T}\}]$ are multivariate polynomials over $\mathbb{Z}_p$, whereas $\xi_{1,i}$, $\xi_{T,i}$ are bit-strings in the encoding sets Ξ (of $\mathbb{G}$) and Ξ_T (of $\mathbb{G}_T$), respectively. Intuitively, the polynomials in lists $\mathcal{L}$ and $\mathcal{L}_T$ correspond to elements of $\mathbb{G}$ and $\mathbb{G}_T$, respectively, that $\mathcal{A}$ will ever be able to compute or guess. In order to simplify the description, we view $\mathbb{Z}_p[\mathsf{X}, \mathsf{X}_0, \mathsf{X}_1, \{\mathsf{U}\}, \{\mathsf{T}\}]$ as a subring of $\mathbb{Z}_p[\mathsf{X}, \mathsf{X}_0, \mathsf{X}_1, \{\mathsf{U}\}, \{\mathsf{V}\}, \{\mathsf{T}\}]$.

Initially, $\tau = 0, \tau_1 = 2q_\Omega + q_g + 3, \tau_T = q_{g_T} + 1$,

$$\begin{aligned}
\mathcal{L} = \{\ & (1, \xi_{1,1}),\ (\mathsf{X}_0, \xi_{1,2}),\ (\mathsf{X}_1, \xi_{1,3}),\ \{(\mathsf{U}_i, \xi_{1,i+3}) : 1 \le i \le q_g\}, \\
& \{(\mathsf{X} + (\mathsf{X}_0 + m_i\mathsf{X}_1)\mathsf{T}_i,\ \xi_{1,2i+q_g+2}),\ (\mathsf{T}_i, \xi_{1,2i+q_g+3}) : 1 \le i \le q_\Omega\}\ \}, \\
\mathcal{L}_T = \{\ & X,\ \{(\mathsf{V}_i, \xi_{T,i+1}) : 1 \le i \le q_{g_T}\}\ \}.
\end{aligned}$$

The bit-strings $\xi_{1,i}$, $\xi_{T,i}$ are set to random distinct strings from Ξ and Ξ_T, respectively. We assume that there is some ordering (say, lexicographic ordering) among the strings in the sets Ξ and Ξ_T, so that given a string $\xi_{1,i}$ or $\xi_{T,i}$, it is possible to determine its index in the lists, if it exits.

The initial state of the two lists correspond to the group elements that $\mathcal{A}$ gets as input as part of the public parameters and the signatures obtained by $\mathcal{A}$ on the messages m_i of its choice.

Remark 3.1 We have assumed above that the adversary will obtain at once (non-adaptively) all the signatures on the messages of its choice, right at the beginning of the game. This is w.l.o.g. since $\mathcal{B}$ provides $\mathcal{A}$ a pair of distinct encodings (in Ξ) as signature for every signature query. Hence $\mathcal{A}$ does not get any added advantage if it is allowed to make signature queries adaptively. Note that this assumption is made only to simplify the description of the proof.

As previously mentioned, the polynomials U_i, V_i correspond to the group elements that $\mathcal{A}$ will guess in the actual interaction. Even in this case we can assume w.l.o.g. that the guesses are made non-adaptively. Since $\mathcal{A}$ can

query the group oracles with at most two new (guessed) elements and since it may also output at most two new elements from $\mathbb{G}$ as its forgery, we have $q_g + q_{g_T} \leq 2q_{\mathcal{O}} + 2$. Hence (3.5) can be simplified as (assuming $q_\Omega \geq 6$, without loss of generality)

$$\tau_1 + \tau_T \leq q_{\mathcal{O}} + 2q_\Omega + 2q_{\mathcal{O}} + 2 + 4 \leq 3(q_{\mathcal{O}} + q_\Omega) \leq 3q. \tag{3.6}$$

The game begins by $\mathcal{B}$ providing $\mathcal{A}$ with the initial τ_1 strings $\xi_{1,1}, \ldots, \xi_{1,\tau_1}$ from $\mathcal{L}$, and τ_T strings $\xi_{T,1}, \ldots, \xi_{T,\tau_T}$ from $\mathcal{L}_T$.

Group operation: The calls made by $\mathcal{A}$ to the group oracles $\mathcal{O}$ and $\mathcal{O}_T$ are modeled as follows. For group operations in $\mathbb{G}$, $\mathcal{A}$ provides $\mathcal{B}$ with two operands (bit-strings) $\xi_{1,i}, \xi_{1,j}$ $(1 \leq i, j \leq \tau_1)$ in $\mathcal{L}$ and also specifies whether to multiply or divide them. $\mathcal{B}$ answers the query by first incrementing the counters $\tau_1 := \tau_1 + 1$ and $\tau := \tau + 1$, and provides $\mathcal{A}$ with the polynomial $\mathsf{F}_{1,\tau_1} := \mathsf{F}_{1,i} \pm \mathsf{F}_{1,j}$. If $\mathsf{F}_{1,\tau_1} = \mathsf{F}_{1,k}$ for some $k < \tau_1$, then $\mathcal{B}$ sets $\xi_{1,\tau_1} := \xi_{1,k}$. Otherwise, ξ_{1,τ_1} is set to a random string distinct from those already present in $\mathcal{L}$. Also the pair $(\mathsf{F}_{1,\tau_1}, \xi_{1,\tau_1})$ is appended to $\mathcal{L}$. Note that the (total) degree of the polynomials $\mathsf{F}_{1,i}$ in $\mathcal{L}$ is at most two. Similarly, group operations in $\mathbb{G}_T$ are answered, appropriately updating the list $\mathcal{L}_T$ and the counters τ_T and τ.

Pairing: For a pairing operation, $\mathcal{A}$ queries $\mathcal{B}$ with two operands $\xi_{1,i}, \xi_{1,j}$ $(1 \leq i, j \leq \tau_1)$ in $\mathcal{L}$. $\mathcal{B}$ first increments $\tau_T := \tau_T + 1$ and $\tau := \tau + 1$, and then computes the polynomial $\mathsf{F}_{T,\tau_T} := \mathsf{F}_{1,i} \cdot \mathsf{F}_{1,j}$. Again, if $\mathsf{F}_{T,\tau_1} = \mathsf{F}_{T,k}$ for some $k < \tau_T$, then $\mathcal{B}$ sets $\xi_{T,\tau_T} := \xi_{T,k}$. Otherwise, ξ_{T,τ_T} is set to a random string distinct from those already present in $\mathcal{L}_T$. Also the pair $(\mathsf{F}_{T,\tau_T}, \xi_{T,\tau_T})$ is appended to $\mathcal{L}_T$. The degree of the polynomials $\mathsf{F}_{T,i}$ in $\mathcal{L}_T$ is at most four.

When $\mathcal{A}$ terminates it outputs $(m, (\xi_{1,\alpha_1}, \xi_{1,\alpha_2})) \in \mathbb{Z}_p \times \mathcal{L} \times \mathcal{L}$ $(1 \leq \alpha_1, \alpha_2 \leq \tau_1)$. This corresponds to the "forgery" output by $\mathcal{A}$ in the actual interaction. Let the polynomials corresponding to ξ_{1,α_1} and ξ_{1,α_2} in $\mathcal{L}$ be F_{1,α_1} and F_{1,α_2}, respectively. After $\mathcal{A}$ terminates, $\mathcal{B}$ computes the polynomial

$$\mathsf{F}_{1,\sigma} := \mathsf{X} + \mathsf{F}_{1,\alpha_2}(\mathsf{X}_0 + mX_1) - \mathsf{F}_{1,\alpha_1}. \tag{3.7}$$

Note that the degree of $\mathsf{F}_{1,\sigma}$ is at most three. Next, $\mathcal{B}$ chooses random values x, x_0, x_1, $\{u\}$, $\{v\}$, $\{t\} \leftarrow \mathbb{Z}_p$ for the indeterminates X, X_0, X_1, $\{\mathsf{U}\}$, $\{\mathsf{V}\}$, $\{\mathsf{T}\}$, respectively. Then it evaluates the polynomials in lists $\mathcal{L}$ and $\mathcal{L}_T$. $\mathcal{A}$ is said to have won the game $\mathcal{G}$ if:

1. $\mathsf{F}_{1,i}(x, x_0, x_1, \{u\}, \{t\}) = \mathsf{F}_{1,j}(x, x_0, x_1, \{u\}, \{t\})$ in $\mathbb{Z}_p$, for some two polynomials $\mathsf{F}_{1,i} \neq \mathsf{F}_{1,j}$ in $\mathcal{L}$.
2. $\mathsf{F}_{T,i}(x, x_0, x_1, \{u\}, \{v\}, \{t\}) = \mathsf{F}_{T,j}(x, x_0, x_1, \{u\}, \{v\}, \{t\})$ in $\mathbb{Z}_p$, for some two polynomials $\mathsf{F}_{T,i} \neq \mathsf{F}_{T,j}$ in $\mathcal{L}_T$.
3. $\mathsf{F}_{1,\sigma}(x, x_0, x_1, \{u\}, \{t\}) = 0$ in $\mathbb{Z}_p$, and $m \neq m_i$ $\forall i$, $i = 1, \ldots, q_\Omega$.

This completes the description of the game $\mathcal{G}$.

We claim that the success probability of $\mathcal{A}$ in the actual EUF-CMA game is bounded above by its success probability in the above game $\mathcal{G}$. This is because of the following reasons (also observed in the proof of Theorem 2.3):

- The conditions 1 and 2 above ensure that $\mathcal{A}$ will get to see only distinct group elements in the actual interaction. In other words, $\mathcal{A}$ is unable to cause *collisions* among group elements. As long as these two conditions are not satisfied, then the view of $\mathcal{A}$ is identical in the game $\mathcal{G}$ and the actual interaction. Hence if $\mathcal{A}$ is unable to provoke collisions, then adaptive strategies are no more powerful than non-adaptive ones (for more details, we refer to [Mau05, Lemma 2 on pp. 12], also [Sho97]). This observation allows us to choose group elements and their representations independently of the strategy of $\mathcal{A}$. Hence $\mathcal{A}$ specified the messages m_i at the beginning of the game $\mathcal{G}$ and also obtained the corresponding signatures (cf. Remark 3.1). For the same reason, it also decided at the beginning itself on the representations it would guess. Note that the assumption that $\mathcal{A}$ would a priori decide the representations it would guess is only to simplify the description of the proof and it is not an inherent limitation.
- The condition 3 above ensures that the pair $(\xi_{1,\alpha_1}, \xi_{1,\alpha_2})$ is a valid forgery on a distinct message m.

We now compute the success probability of $\mathcal{A}$ in the game $\mathcal{G}$. The τ_1 polynomials $\mathsf{F}_{1,i}$ in $\mathcal{L}$ have degree at most two. Note that $\mathsf{F}_{1,i} \neq \mathsf{F}_{1,j} \Leftrightarrow \mathsf{F}_{1,i} - \mathsf{F}_{1,j} \neq 0$ as polynomials. From Lemma 2.3 (with $\lambda' = 0$), the probability that two distinct polynomials in $\mathcal{L}$ evaluate to the same value for randomly and independently chosen values for the indeterminates is at most $\frac{2}{p}$. Summing up over at most $\binom{\tau_1}{2}$ distinct pairs (i, j), the probability that the condition 1 above holds is at most $\binom{\tau_1}{2} \cdot \frac{2}{p}$. Similarly, we have the probability that the condition 2 above holds is at most $\binom{\tau_2}{2} \cdot \frac{4}{p}$. The degree of the polynomial $\mathsf{F}_{1,\sigma}$ in condition 3 is at most three. In order to apply Lemma 2.3, we need to prove that F_σ is not identically equal to the zero polynomial. We prove this fact in Lemma 3.1 below. Let $\Pr^{forge}_{\mathcal{A},\Pi_{\mathsf{BB}}}$ denote the advantage of the adversary $\mathcal{A}$ in computing a forgery against Π_{BB}. Then, assuming Lemma 3.1, we obtain from (3.6)

$$\Pr^{forge}_{\mathcal{A},\Pi_{\mathsf{BB}}} \leq \binom{\tau_1}{2} \cdot \frac{2}{p} + \binom{\tau_2}{2} \cdot \frac{4}{p} + \frac{3}{p} \leq \frac{2}{p}(\tau_1 + \tau_2)^2 \leq \frac{18q^2}{p}. \tag{3.8}$$

Hence if $q = poly(\log p)$, then $\Pr^{forge}_{\mathcal{A},\Pi_{\mathsf{BB}}}$ is negligible. □

Lemma 3.1. *The polynomial* $\mathsf{F}_{1,\sigma} \in \mathbb{Z}_p[\mathsf{X}, \mathsf{X}_0, \mathsf{X}_1, \{\mathsf{U}\}, \{\mathsf{T}\}]$ *is non-zero.*

Proof. Any polynomial in $\mathcal{L}$ is obtained by either adding or subtracting two polynomials previously existing in the list. Hence we can write F_{1,α_1} and F_{1,α_2} in terms of polynomials present in $\mathcal{L}$ when it was was initialized at step $\tau = 0$

in the game $\mathcal{G}$. Note that initially $\mathcal{L}$ also includes the representations guessed by $\mathcal{A}$, in addition to the inputs.

$$\begin{aligned}\mathsf{F}_{1,\alpha_1} = \quad & c_1 + c_2\mathsf{X}_0 + c_3\mathsf{X}_1 + \textstyle\sum_{i=1}^{q_g} c_{4,i}\mathsf{U}_i + \sum_{i=1}^{q_\Omega} c_{5,i}\mathsf{T}_i \\ & + \textstyle\sum_{i=1}^{q_\Omega} c_{6,i}(\mathsf{X} + (\mathsf{X}_0 + m_i\mathsf{X}_1)\mathsf{T}_i),\end{aligned} \tag{3.9}$$

$$\begin{aligned}\mathsf{F}_{1,\alpha_2} = \quad & d_1 + d_2\mathsf{X}_0 + d_3\mathsf{X}_1 + \textstyle\sum_{i=1}^{q_g} d_{4,i}\mathsf{U}_i + \sum_{i=1}^{q_\Omega} d_{5,i}\mathsf{T}_i \\ & + \textstyle\sum_{i=1}^{q_\Omega} d_{6,i}(\mathsf{X} + (\mathsf{X}_0 + m_i\mathsf{X}_1)\mathsf{T}_i),\end{aligned} \tag{3.10}$$

where $c_j, d_j (j = 1, 2, 3), c_{j,i}, d_{j,i} (j = 4, 5, 6; 1 \leq i \leq q_\Omega) \in \mathbb{Z}_p$ are chosen by $\mathcal{A}$. We have two possible cases:

Case 1: $c_{6,i} = d_{6,i} = 0 \quad \forall i,\ 1 \leq i \leq q_\Omega$.

In this case, both F_{1,α_1} and F_{1,α_2} do not contain the indeterminate X. Hence the expression $\mathsf{F}_{1,\alpha_2}(\mathsf{X}_0 + m\mathsf{X}_1) - \mathsf{F}_{1,\alpha_1}$ in (3.7) is free of X. Therefore, in the polynomial $\mathsf{X} + \mathsf{F}_{1,\alpha_2}(\mathsf{X}_0 + m\mathsf{X}_1) - \mathsf{F}_{1,\alpha_1}$, the coefficient of the term X is non-zero. Hence $\mathsf{F}_{1,\sigma}$ is non-zero.

Case 2: $c_{6,k} \neq 0$ or $d_{6,k} \neq 0$ for some k, where $1 \leq k \leq q_\Omega$.

On substituting expressions from (3.9) and (3.10) into (3.7), we get that the coefficient of monomials $\mathsf{X}_0^2\mathsf{T}_i$, $\mathsf{X}_0\mathsf{T}_i$, $\mathsf{X}_1\mathsf{T}_i$ in $\mathsf{F}_{1,\sigma}$ are $d_{6,i}$, $d_{5,i} - c_{6,i}$, $m\,d_{5,i} - m_i c_{6,i}$, respectively, for $1 \leq i \leq q_\Omega$.

If $d_{6,k} \neq 0$, then the coefficient of $\mathsf{X}_0^2\mathsf{T}_k$ is non-zero, and hence $\mathsf{F}_{1,\sigma} \neq 0$. Else, $c_{6,k} \neq 0$. We again have two cases: If $d_{5,k} \neq c_{6,k}$, then the coefficient of $\mathsf{X}_0\mathsf{T}_k$ is non-zero. Or else, if $d_{5,k} = c_{6,k}$, then the coefficient of $\mathsf{X}_1\mathsf{T}_k$ is non-zero, since $m \neq m_i\ \forall i, i = 1, \ldots, q_\Omega$. Hence in all cases we have $\mathsf{F}_{1,\sigma}$ to be a non-zero polynomial. □

3.4 A Leakage-Resilient Boneh–Boyen Signature Scheme

As previously mentioned in Section 3.2.1, any cryptographic scheme that does not maintain a stateful secret state is insecure against continual leakage. So is the case with the signature scheme Π_{BB}. We now describe a leakage-resilient version Π^*_{BB} of Π_{BB}. We follow the techniques of [KP10a] to adapt Π_{BB} to a leakage setting. The basic idea is to store the secret key $X = g^x$ in two different parts of the memory as $(S_0 := g^{l_0}, S'_0 := g^{x-l_0})$ for a randomly chosen $l_0 \leftarrow \mathbb{Z}_p$. Accordingly, the $\mathsf{KeyGen}_{\mathsf{BB}}$ step of Π_{BB} is modified to obtain the set-up stage $\mathsf{KeyGen}^*_{\mathsf{BB}}$ of Π^*_{BB}. The signature generation is now carried out as a two step process $\mathsf{Sign}^*_{\mathsf{BB1}}$ and $\mathsf{Sign}^*_{\mathsf{BB2}}$. During the i^{th} signature query, the two parts of the secret key (S_{i-1}, S'_{i-1}) are refreshed to obtain $(S_i := S_{i-1} \cdot g^{l_i}, S'_i := S'_{i-1} \cdot g^{-l_i})$, where $l_i \leftarrow \mathbb{Z}_p$. This is done in order to protect against continual leakage.

Let $\Pi^*_{\mathsf{BB}} = (\mathsf{KeyGen}^*_{\mathsf{BB}}, \mathsf{Sign}^*_{\mathsf{BB1}}, \mathsf{Sign}^*_{\mathsf{BB2}}, \mathsf{Verify}^*_{\mathsf{BB}})$ be a stateful signature scheme on the message space $\mathbb{Z}_p$ defined as follows:

1. $\mathsf{KeyGen}^*_{\mathsf{BB}}(\kappa)$: Compute $\mathbb{PP} := (\mathbb{G}, \mathbb{G}_T, p, e, g) \leftarrow \mathsf{BGen}(\kappa)$. Choose random $x, x_0, x_1, l_0 \leftarrow \mathbb{Z}_p$. Set $X := g^x$, $X_0 := g^{x_0}$, $X_1 := g^{x_1}$ and $X_T := e(g, X) = e(g, g)^x$. The public key is $pk := (\mathbb{PP}, X_0, X_1, X_T)$ and the secret key is $sk^* := (S_0 := g^{l_0},\ S'_0 := g^{x-l_0} = X \cdot g^{-l_0}) \in \mathbb{G}^2$.
2. $\mathsf{Sign}^*_{\mathsf{BB1}}(S_{i-1}, m_i)$: Choose random $t_i, l_i \leftarrow \mathbb{Z}_p$. Set $S_i := S_{i-1} \cdot g^{l_i}$, $\sigma'_{1,i} := S_i \cdot (X_0 \cdot X_1^{m_i})^{t_i}$, and $\sigma'_{2,i} := g^{t_i}$.
3. $\mathsf{Sign}^*_{\mathsf{BB2}}(S'_{i-1}, (\sigma'_{1,i}, \sigma'_{2,i}, l_i))$: Set $S'_i := S'_{i-1} \cdot g^{-l_i}$ and $\sigma_i := (S'_i \cdot \sigma'_{1,i}, \sigma'_{2,i})$. Output the signature σ_i.
4. $\mathsf{Verify}^*_{\mathsf{BB}}(pk, m, \sigma)$: Let $\sigma = (\sigma_1, \sigma_2) \in \mathbb{G}^2$. Output the bit $b = 1$ (*valid*) if $X_T \star e(\sigma_2, X_0 \cdot X_1^m) = e(\sigma_1, g)$. Otherwise output $b = 0$ (*invalid*).

In steps 2 and 3 above, the index i keeps a count of the number of invocations (rounds) of the signing algorithm. For every $i \geq 1$, let $Y_i := \sum_{j=0}^{i} l_j$. It is easy to check that $S_i \cdot S'_i = g^{Y_i} \cdot g^{x-Y_i} = X$. We sometimes even refer to X as the secret key.

Note that $\mathsf{Sign}^*_{\mathsf{BB1}}$ requires four exponentiations and $\mathsf{Sign}^*_{\mathsf{BB2}}$ requires one. The total number of exponentiations needed for every signature invocation can be reduced from five to four if $\mathsf{Sign}^*_{\mathsf{BB1}}$ also passes on g^{l_i} to $\mathsf{Sign}^*_{\mathsf{BB2}}$. Hence only one extra exponentiation is needed when compared with the $\mathsf{Sign}_{\mathsf{BB}}$ step of Π_{BB}, which requires three.

For the sake of clarity, we would like to compare the various notations used in the signature scheme Π^*_{BB} above with those in (3.1) corresponding to a generic stateful signature scheme Π^*. The quantities r_i and w_i in (3.1) correspond to (l_i, t_i) and $(\sigma'_{1,i}, \sigma'_{2,i}, l_i)$ of Π^*_{BB}, respectively. The quantities S_i, S'_i and m_i denote the same things in both the cases. However, since the algorithm $\mathsf{Sign}^*_{\mathsf{BB2}}$ of Π^*_{BB} does not generate any randomness, there is no analogue in Π^*_{BB} for r'_i of (3.1). Accordingly, the leakage functions specified by an adversary to the signing oracle $\Omega^{\mathrm{leak}}_{(S_{i-1}, S'_{i-1})}(m_i, f_i, h_i)$ would be of the form $f_i(S_{i-1}, (l_i, t_i))$ and $h_i(S'_{i-1}, (\sigma'_{1,i}, \sigma'_{2,i}, l_i))$.

First we show that Π^*_{BB} is secure in the GBG model when an adversary is not allowed to obtain leakage. The following lemma is a trivial consequence of the fact that the input/output behaviour of Π^*_{BB} and Π_{BB} are identical (c.f. Theorem 3.1).

Lemma 3.2. *The signature scheme* Π^*_{BB} *is* EUF-CMA secure *in the generic bilinear group model.*

The following theorem establishes the fact that the signature scheme Π^*_{BB} is resilient to (continual) leakage attacks in the GBG model if $\lambda \ll \frac{\log p}{2}$, where λ is the leakage parameter.

Theorem 3.2. *The signature scheme* Π^*_{BB} *is secure with leakage w.r.t. Definition 3.2 in the generic bilinear group model. The advantage of a q-query adversary who gets at most* λ *bits of leakage per each invocation of* $\mathsf{Sign}^*_{\mathsf{BB1}}$ *or* $\mathsf{Sign}^*_{\mathsf{BB2}}$ *is* $O\left(\frac{q^2}{p}2^{2\lambda}\right)$.

Proof. Let $\mathcal{A}$ be a q-query adversary that can break the security of Π^*_{BB}. By a q-query adversary $\mathcal{A}$ we mean that $\mathcal{A}$ can make totally at most q group oracle and signing oracle queries. Let $q_{\mathcal{O}}$ be the total number of calls to the group oracles $\mathcal{O}$, $\mathcal{O}_T$ and $\mathcal{O}_e$, and q_Ω correspond to the number of calls to the signing oracle. We have $q_{\mathcal{O}} + q_\Omega \leq q$. In the count $q_{\mathcal{O}}$, even the group oracle queries by leakage functions f_i, h_i specified by $\mathcal{A}$ are also included.

We first informally sketch the main ideas of the proof and then formalize these ideas. Let us try to see why the proof of security of Π^*_{BB} in the absence of any leakage (i.e. proof of Theorem 3.1) would not carry over as it is in the presence of leakage. In the non-leakage setting, while determining the probability of collision among distinct polynomials in conditions 1-3 on page 55, we substituted for each indeterminate an independent value chosen from an uniform distribution over $\mathbb{Z}_p$. But, when $\mathcal{A}$ has access to leakage functions $f_i(S_{i-1}, (l_i, t_i))$ and $h_i(S'_{i-1}, (\sigma'_{1,i}, \sigma'_{2,i}, l_i))$, then from its point of view the parameters t_i $(1 \leq i \leq q_\Omega)$ are no longer uniformly distributed (though they are still independent). With some partial information about t_i, $\mathcal{A}$ can now cause collisions among polynomials with increased probability. Since each t_i is chosen independently and it can be leaked by only f_i, hence at most λ bits of t_i can be leaked. Apart from the values t_i, the only other "useful" information that leakage functions can provide is about the secret key $X = g^x$. This is because the parameters l_i themselves alone do not help $\mathcal{A}$ to output forgery since the signatures generated are independent of these randomly chosen values. Instead, $\mathcal{A}$ can very much use the leakages of l_i to compute, and eventually leak, the secret key X. Note that the leakage functions do not provide any additional information on the values x, x_0 or x_1.

We first bound the probability of the event that the secret key X is computed by some leakage function f_i or h_i. As long as this event has not occurred, then no bits of the secret key is leaked and the "only" additional information $\mathcal{A}$ has is about the values t_i. Clearly, the probability of this event depends on the leakage parameter λ. For instance, if the amount of leakage per invocation is not bounded, then during the first signature query itself, the adversary can leak the initial two shares of the secret key $S_0 = g^{l_0}$ and $S'_0 = X \cdot g^{-l_0}$ to recompute X. Finally, we determine the advantage of $\mathcal{A}$ conditioned on the event of the secret key X not being computed by any of the leakage functions.

Formally, we define E to be the event of computing (or guessing) the secret key $X = g^x$ by any of the leakage functions f_i or h_i $(1 \leq i \leq q_\Omega)$. Let $\overline{E}$ denote the complement of the event E, *Forgery* denote the event of $\mathcal{A}$ forging a signature on a new message, and $\Pr^{forge}_{\mathcal{A},\Pi^*_{\mathsf{BB}}} = \Pr[\text{Forgery}]$ denote the advantage

of $\mathcal{A}$ in computing a forgery against Π^*_{BB}. We have

$$\Pr^{forge}_{\mathcal{A},\Pi^*_{\mathsf{BB}}} = \Pr[\text{Forgery}|E]\Pr[E] \;+\; \Pr[\text{Forgery}|\overline{E}]\Pr[\overline{E}].$$

Since $\Pr[\text{Forgery}|E]$, $\Pr[\overline{E}] \leq 1$, we obtain

$$\Pr^{forge}_{\mathcal{A},\Pi^*_{\mathsf{BB}}} \leq \Pr[E] + \Pr[\text{Forgery}|\overline{E}]. \tag{3.11}$$

We first bound the probability of the event E.

Lemma 3.3. $\Pr[E] \leq O\left(\frac{q^2}{p}2^{2\lambda}\right)$.

Proof. Let the adversary $\mathcal{A}$ play the following game $\mathcal{G}'$. Since the game $\mathcal{G}'$ is similar in nature to the game $\mathcal{G}$ in the proof of Theorem 3.1, we only briefly describe $\mathcal{G}'$. We use the notations introduced in the game $\mathcal{G}$. Let $\{\mathsf{L}\}$ denote the list of indeterminates $\{\mathsf{L}_i : 1 \leq i \leq q_\Omega\}$ that correspond to the values l_i in Π^*_{BB}.

Game $\mathcal{G}'$: For every leakage function $f_i(S_{i-1}, (l_i, t_i))$ and $h_i(S'_{i-1}, (\sigma'_{1,i}, \sigma'_{2,i}, l_i))$, $\mathcal{A}$ builds lists $\mathcal{L}^{f_i}$ and $\mathcal{L}^{h_i}$, respectively. These lists contain polynomial-bit string pairs. The polynomials are from $\mathbb{Z}_p[\mathsf{X}, \mathsf{X}_0, \mathsf{X}_1, \{\mathsf{U}\}, \{\mathsf{T}\}, \{\mathsf{L}\}]$ and the bit-strings are from the encoding set Ξ of the group $\mathbb{G}$. Intuitively, the polynomials in lists $\mathcal{L}^{f_i}$ and $\mathcal{L}^{h_i}$ correspond to the elements of group $\mathbb{G}$ that can be computed by f_i and h_i, respectively. Every polynomial in $\mathcal{L}^{f_i}$ is of the form

$$c_{1,i}\mathsf{L}_i + c_{2,i}\sum_{j=0}^{i-1}\mathsf{L}_j + c_{3,i}\mathsf{D}_i, \tag{3.12}$$

where $c_{1,i}$, $c_{2,i}$, $c_{3,i} \in \mathbb{Z}_p$ are chosen by $\mathcal{A}$ and $D_i \in \mathbb{Z}_p[\mathsf{X}, \mathsf{X}_0, \mathsf{X}_1, \{\mathsf{U}\}, \{\mathsf{T}\}]$ is in the list $\mathcal{L}$ (c.f. (3.3)). Every polynomial in $\mathcal{L}^{h_i}$ is of the form

$$d_{1,i}\mathsf{L}_i + d_{2,i}\left(\mathsf{X} - \sum_{j=0}^{i-1}\mathsf{L}_j\right) + d_{3,i}\left(\left(\sum_{j=0}^{i}\mathsf{L}_j\right) + (\mathsf{X}_0 + m_i\mathsf{X}_1)\mathsf{T}_i\right) + d_{4,i}\mathsf{W}_i, \tag{3.13}$$

where $d_{1,i}$, $d_{2,i}$, $d_{3,i}$, $d_{4,i}$, $m_i \in \mathbb{Z}_p$ are also chosen by $\mathcal{A}$ and $\mathsf{W}_i \in \mathbb{Z}_p[\mathsf{X}, \mathsf{X}_0, \mathsf{X}_1, \{\mathsf{U}\}, \{\mathsf{T}\}]$ is in the list $\mathcal{L}$.

When $\mathcal{A}$ terminates it outputs a polynomial F from the list $\mathcal{L}^{f_i}$ or $\mathcal{L}^{h_i}$, for some i. Intuitively, the polynomial F output by $\mathcal{A}$ corresponds to its guess of the secret key X. $\mathcal{A}$ is said to have won the game $\mathcal{G}'$ if

1. There is a collision in any of the lists $\mathcal{L}^{f_i}$ and $\mathcal{L}^{h_i}$, for some i $(1 \leq i \leq q_\Omega)$.
2. $\mathsf{F} - \mathsf{X} = 0$ in $\mathbb{Z}_p$.

Note that the polynomials are now evaluated with values chosen from independent distributions with min-entropy $\log p - 2\lambda$. The reason for this will be shortly explained. This completes the description of the game $\mathcal{G}'$.

Technically speaking, $\mathcal{A}$ must also maintain lists $\mathcal{L}_T^{f_i}$ and $\mathcal{L}_T^{h_i}$ $(1 \leq i \leq q_\Omega)$ that correspond to elements of the group $\mathbb{G}_T$ that can be computed by f_i and h_i. To simplify the discussion, we only describe collisions in the lists $\mathcal{L}^{f_i}$ and $\mathcal{L}^{h_i}$. Similar arguments apply for the lists $\mathcal{L}_T^{f_i}$ and $\mathcal{L}_T^{h_i}$. Since we compute $\Pr[E]$ only up to a constant factor, the additional advantage $\mathcal{A}$ obtains from collisions in $\mathcal{L}_T^{f_i}$ and $\mathcal{L}_T^{h_i}$ is implicitly included. However, working on the lines of the proof of Theorem 3.1, it is relatively straightforward to completely formalize the present discussion.

For similar reasons as given in the proof of Theorem 3.1, we have $\Pr[E]$ is bounded above by the success probability of $\mathcal{A}$ in the above game $\mathcal{G}'$. We particularly like to note the following. As observed in [AM11, pp. 691] and the references therein, even in the leakage setting adaptive strategies are no more powerful than non-adaptive ones.

Before computing the success probability of $\mathcal{A}$, we first show that $\mathsf{F} - \mathsf{X}$ is a non-zero polynomial. From Lemma 3.2 and Theorem 3.1, we know that Π_{BB}^* is secure without leakage. Hence the polynomial X (that corresponds to the secret key) cannot appear in the list $\mathcal{L}$, because this would otherwise imply that the secret key can be computed without access to leakage functions. A formal proof for this fact can be easily obtained on the lines of the proof of Lemma 3.1. Hence even when $c_{1,i} = c_{2,i} = 0$ in (3.12), the lists $\mathcal{L}^{f_i}$ cannot contain the polynomial X. If $c_{1,i} \neq 0$ or $c_{2,i} \neq 0$, then the polynomial in (3.12) will contain either L_i or L_{i-1}, or both. Hence the polynomial X cannot appear in any of the lists $\mathcal{L}^{f_i}$. In a similar way it can be seen that the lists $\mathcal{L}^{h_i}$ do not contain X. Hence $\mathsf{F} - \mathsf{X}$ is a non-zero polynomial of degree at most two.

Let us now determine the probability that the condition 1 above holds, i.e. the probability of collisions among distinct polynomials in any of the lists $\mathcal{L}^{f_i}$ and $\mathcal{L}^{h_i}$. In order to compute the probability, we evaluate the polynomials in (3.12) and (3.13) by choosing values from $\mathbb{Z}_p$ according to (independent) distributions with min-entropy at least $\log p - 2\lambda$. This is because $\mathcal{A}$ can obtain at most 2λ bits of leakage about l_i $(i = 0, \ldots, q_\Omega)$, and at most λ bits of t_i $(i = 1, \ldots, q_\Omega)$. From Lemma 2.1, the values l_i, t_i have min-entropy at least $\log p - 2\lambda$ in the view of $\mathcal{A}$. The total length of the lists $\mathcal{L}^{f_i}$, $\mathcal{L}^{h_i}$ is at most $O(q_\Omega + q_\mathcal{O}) = O(q)$. Hence there can be at most $O(q^2)$ pairs of distinct polynomials (of degree at most two) evaluating to the same value. From Lemma 2.3 (with $\lambda' = 2\lambda$), we obtain $\Pr[E] \leq O\left(\frac{q^2}{p} 2^{2\lambda}\right)$. Since $\mathsf{F} - \mathsf{X}$ is a non-zero polynomial of degree at most two, the probability that $\mathsf{F} - \mathsf{X}$ evaluates to zero is at most $\frac{2}{p} 2^{2\lambda}$. This probability is also implicitly included in the above bound.

□

We now determine the probability $\Pr[\text{Forgery} \mid \overline{E}]$ in (3.11).

Lemma 3.4. $\Pr[\text{Forgery} \mid \overline{E}] \leq \dfrac{18q^2}{p} 2^{\lambda}$.

Proof. Given that the event E has not occurred, the only meaningful leakage $\mathcal{A}$ can now obtain is that of t_i ($i = 1, \ldots, q_\Omega$). Since at most λ bits of t_i can leak (only by f_i), from the view point of $\mathcal{A}$ the values t_i have min-entropy at least $\log p - \lambda$. From Lemma 2.3 (with $\lambda' = \lambda$), the probability of collision among distinct polynomials in conditions 1-3 on page 55 is now increased by a factor of 2^{λ}. Hence, from (3.8), we obtain $\Pr[\text{Forgery} \mid \overline{E}] \leq \frac{18q^2}{p} 2^{\lambda}$. □

From (3.11) and Lemmas 3.3 and 3.4, we have $\Pr^{forge}_{\mathcal{A},\Pi^*_{\mathsf{BB}}} \leq O\left(\dfrac{q^2}{p} 2^{2\lambda}\right)$. This completes the proof of Theorem 3.2. □

3.5 Conclusion and Future Directions

In this chapter, we proposed a leakage-resilient Boneh–Boyen signature scheme in the continual split-state leakage model that tolerates approximately half the bits of (a share of) the secret key at every invocation. We proved the security of our construction in the generic bilinear group model. To generate a signature, the number of exponentiations needed in the pairing base group is four (cf. Section 3.4). Hence we require only one extra exponentiation compared to the original Boneh–Boyen signature scheme. We also observed that the Boneh–Boyen signature scheme is existentially unforgeable in the GBG model, whereas it is only known to be selectively unforgeable in the standard model.

It seems that our results readily extend to the min-entropy leakage model of Chapter 2. Also it appears straightforward to extend our results to the full Boneh–Boyen identity-based encryption scheme. We also expect it to be fully secure (against adaptive identity attacks) in the GBG model, while it is only known to be secure against selective identity attacks in the standard model.

An interesting direction would be to obtain efficient leakage-resilient signature schemes that tolerate continual leakage in the full memory leakage model where the entire secret state is input to the leakage functions. With the current techniques we expect that such schemes, if they exist, can possibly be proven secure only in an idealized model of computation, such as the generic group model or the random oracle model.

Chapter 4

Split-State Pairing-based Schnorr Signature Scheme

In this chapter, we propose a pairing analogue of the classical Schnorr signature scheme. We next transform it to include split signing key updates, similar to what was done in the earlier chapters. We give a leakage-resilience bound in the generic bilinear group model against continual split-state leakage attacks for the new scheme. Our scheme tolerates leakage of almost half of the bits of the secret key at every new signature invocation. The secret key's storage space is constant and it is uniquely determined by the public key, the properties also enjoyed by schemes in the previous chapters.

Contents

4.1 Introduction

We aim at building a signature scheme secure against continual leakage in the split-state model that builds on the Schnorr signature scheme [Sch91]. Apart from being of possible interest to the cryptographic engineering community due to its efficiency, it exhibits certain properties of theoretical interest as well. Notice that several works [Kat09, ADW09, FKPR10] have already built leakage-resilient signature schemes based on Schnorr. All of these works confirm the finding by Wichs [Wic13], who seems to indicate that it might be impossible to achieve continual leakage-resilience for cryptosystems whose secret key is uniquely determined by its public key, unless we weaken the security model. All the above mentioned schemes are built by gluing together

several copies of the basic Schnorr signature scheme (a technique that was first used by Okamoto [Oka92]), and thus given its public key there are exponentially many possible secret keys. The works [Kat09, ADW09] only allow a bounded leakage during the life-time of the protocol, although in their model every part of the memory is susceptible to leak (as opposed to the split-state model); the work [FKPR10] uses the split-state model and allows roughly 1/36 leakage ratio at every signing step, but the number of signature queries is bounded in advance. Our goal is to provide a Schnorr-like signature scheme where the secret key material to be stored is constant at any time, since in the aforementioned works the secret keys' storage is proportional to the leakage ratio allowed. In particular we propose a scheme where the secret key is uniquely determined by its public key, the secret key consists of only two group elements at any given time and it is unforgeable even if the number of adversarial signature queries is not known in advance.

From the results of Section 2.1, we see that there are attacks (in the continual split-state leakage model) for the split-secret key variant of the original Schnorr scheme instantiated over any cryptographic group $\mathbb{G}$ where the discrete logarithm problem is assumed to be hard. This is why we state our theorems with respect to a transposition of the modified Schnorr signature scheme to pairing groups, where the secret key is no longer $x \in \mathbb{Z}_p$ but $X = g^x \in \mathbb{G}$, where $\mathbb{G}$ is the base pairing group with $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. This allows us to use the generic bilinear group (GBG) model that will ease our analysis. We proceed by first showing that our transposition of the Schnorr signature scheme to pairing groups is existentially unforgeable [GMR88] in the GBG model. This is achieved by showing that the security reduction in the generic group model [Sho97] for elliptic-curve based Schnorr signatures recently given by Neven, Smart and Warinschi [NSW09] can be translated to the GBG and allows to deal with data leakage. Secondly, we modify the pairing-based Schnorr scheme by multiplicatively sharing $X = X_1 \cdot X_2$, where $X_1, X_2 \in \mathbb{G}$, and by breaking the signing scheme into two phases, each one using the corresponding share X_1 or X_2. Again, at each new signature invocation a fresh sharing (X_1', X_2') of X is computed. Our main theorem (Theorem 4.2) states that allowing λ bits of leakage at each phase of every round overall decreases the security of the scheme by a factor of at most $2^{2\lambda}$ in our leakage model, which is the same as the one used in Chapter 3. Also, our Schnorr-like scheme has efficiency comparable to that of our scheme from Chapter 3.

In Section 4.2, we introduce a bilinear variant of the Schnorr signature scheme and prove its security (without leakage) in the GBG model. In Section 4.3, we split the secret state of the bilinear Schnorr scheme and prove its leakage resilience under continual split-state leakage in the GBG model.

4.2 Basic Pairing-based Schnorr Signature Scheme

We propose a bilinear variant of the Schnorr signature scheme [Sch89, Sch91].

Let $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p$ be a hash function. The signature scheme $\Pi_{\mathsf{Sc}} = (\mathsf{KeyGen}_{\mathsf{Sc}}, \mathsf{Sign}_{\mathsf{Sc}}, \mathsf{Verify}_{\mathsf{Sc}})$, defined on the message space $\{0,1\}^*$, is as follows:

1. $\mathsf{KeyGen}_{\mathsf{Sc}}(\kappa)$: Compute $\mathbb{PP} := (\mathbb{G}, \mathbb{G}_T, p, e, g) \leftarrow \mathsf{BGen}(\kappa)$. Choose random $x \leftarrow \mathbb{Z}_p$. Set $X := g^x$, $g_T := e(g,g)$, and $X_T := e(g,X) = g_T^x$. The public key is $pk := (\mathbb{PP}, X_T, \mathsf{H})$ and the secret key is $sk := X$.
2. $\mathsf{Sign}_{\mathsf{Sc}}(sk, m)$: Choose a random $t \leftarrow \mathbb{Z}_p$. Set $\gamma := \mathsf{H}(g_T^t || m)$, $Y := g^t \cdot X^\gamma$ and $\sigma := (Y, \gamma)$. Output the signature σ.
3. $\mathsf{Verify}_{\mathsf{Sc}}(pk, m, \sigma)$: Let $\sigma = (Y, \gamma) \in \mathbb{G} \times \mathbb{Z}_p$. Set $\rho := e(Y,g) \star (g_T^x)^{-\gamma}$. Output the bit $b = 1$ (*valid*) if $\mathsf{H}(\rho||m) = \gamma$. Otherwise output $b = 0$ (*invalid*).

We now prove the security of the above scheme in the GBG model relative to two hardness assumptions about the hash function H that were introduced in [NSW09], and which are recalled below. These two assumptions are *weaker* than collision-resistance [NSW09]. We adapt the proof techniques of [NSW09] to the bilinear setting.

Definition 4.1. [Random-Prefix (Second-) Preimage problem [NSW09]] *The advantage of an adversary $\mathcal{A}$ in solving the Random-Prefix Preimage (**RPP**) problem (respectively, Random-Prefix Second-Preimage (**RPSP**) problem) for a hash function $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p$, with prefix in a set of bit-strings D, is given by*

$$\begin{aligned}
\mathsf{Adv}_{\mathsf{H}}^{\mathsf{RPP[D]}}(\mathcal{A}) &= \Pr\left[\mathsf{H}(R||m) = \gamma \ : \ \gamma \leftarrow \mathcal{A}, \text{ random } R \leftarrow D, \, m \leftarrow \mathcal{A}(R)\right], \\
\mathsf{Adv}_{\mathsf{H}}^{\mathsf{RPSP[D]}}(\mathcal{A}) &= \Pr\big[\mathsf{H}(R||m) = \mathsf{H}(R||m') \ : \ m \leftarrow \mathcal{A}, \text{ random } R \leftarrow D, \\
&\qquad m' \leftarrow \mathcal{A}(R), \, m' \neq m\big],
\end{aligned}$$

where the probability is taken over R and the coins of $\mathcal{A}$. The RPP *problem (respectively,* RPSP *problem) for H is said to be (t, ϵ) hard if no adversary $\mathcal{A}$ with running time at most t has advantage greater than ϵ in solving it.*

Theorem 4.1. *The signature scheme Π_{Sc} is* EUF-CMA *secure in the generic bilinear group model if the* RPP$[\Xi_T]$ *and* RPSP$[\Xi_T]$ *problems are hard for H.*

Proof. Let $\mathcal{A}$ be a (Γ-time, q-query) adversary that can break the security of Π_{Sc}. Hence $\mathcal{A}$ can make totally at most q group oracle, pairing oracle and signing oracle queries, and runs in time at most Γ. Let $q_{\mathcal{O}}$ denote the total number of calls to the oracles $\mathcal{O}$, $\mathcal{O}_T$ and $\mathcal{O}_e$, and q_Ω denote the number of calls to the signing oracle Ω_{Sc}. Thus $q_{\mathcal{O}} + q_\Omega \leq q$.

$\Pr^{forge}_{\mathcal{A},\Pi_{\mathsf{Sc}}}$ denote the advantage of the adversary $\mathcal{A}$ in computing a forgery against Π_{Sc}. Also let RPP$[\Xi_T]$ and RPSP$[\Xi_T]$ problems be $(\Gamma', \epsilon_{\mathrm{RPP}})$- and $(\Gamma', \epsilon_{\mathrm{RPSP}})$-hard for the hash function H. We show that

$$\Pr^{forge}_{\mathcal{A},\Pi_{\mathsf{Sc}}} \ \leq \ 2q \cdot \epsilon_{\mathrm{RPSP}} + 36q^2 \cdot \epsilon_{\mathrm{RPP}} + \frac{15q^2}{p} + \frac{108q^3}{p}$$

for any (Γ-time, q-query) adversary $\mathcal{A}$ in the GBG model, where $\Gamma' \approx \Gamma$. More precisely, Γ' is the sum of Γ and the time required by simulator to maintain the environment.

The main idea is to use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that solves both the RPP[Ξ_T] and the RPSP[Ξ_T] problems for the hash function H. $\mathcal{B}$ will simulate EUF-CMA experiment for $\mathcal{A}$ in the naive way, through the game $\mathcal{G}$ that we later describe below. In the game, $\mathcal{B}$ also simulates the generic bilinear group oracles in the usual way by maintaining lists of pairs of encodings and polynomials that represent the relation amongst group elements. Let $\mathcal{C}$ be a challenger trying to prove the hardness of both the RPP[Ξ_T] and the RPSP[Ξ_T] problems for H against $\mathcal{B}$.

There are only two possibilities for $\mathcal{A}$ to output a forgery:

1. $\mathcal{A}$ uses a signature previously obtained to output a forgery (on a distinct message).
2. $\mathcal{A}$ does *not* output a previously obtained signature as a forgery.

Note that in a forgery of type 1, the "random prefix" for the hash function input (during verification) is the same as that for the corresponding previously obtained signature. In this case $\mathcal{B}$ will attempt to solve the RPSP[Ξ_T] problem for H. There are two issues that $\mathcal{B}$ needs to address in this case to solve the RPSP problem. First, it must correctly guess at the beginning of the simulation when $\mathcal{A}$ outputs a forgery of type 1 (and accordingly inform $\mathcal{C}$ that it attempts to solve the RPSP problem). Secondly, $\mathcal{B}$ needs to guess a priori which one of the previous signatures will $\mathcal{A}$ use for the forgery. Then during that step $\mathcal{B}$ needs to forward the corresponding message to the (now RPSP) challenger $\mathcal{C}$ to obtain a random prefix as part of its RPSP challenge. This random prefix will be used as the encoding of the corresponding element g_T^t during the above signing step. $\mathcal{B}$ solves the RPSP problem by forwarding to its (now RPSP) challenger $\mathcal{C}$ the forged message output by $\mathcal{A}$. Note that the probability that $\mathcal{B}$ will succeed in both the guesses is at least $\frac{1}{2q}$.

In the case of forgery of type 2, $\mathcal{B}$ will attempt to solve the RPP[Ξ_T] problem for H. Again, $\mathcal{B}$ must first guess correctly when this type of forgery occurs (and accordingly inform $\mathcal{C}$ that it attempts to solve the RPP problem). Secondly, $\mathcal{B}$ must commit to a value γ to obtain a random prefix $R \in \Xi_T$ as part of a RPP challenge. Eventually when $\mathcal{A}$ outputs a forgery on a (distinct) message m, it must turn out that the encoding of the "corresponding g_T^t" must be R and that $\mathsf{H}(R||m) = \gamma$. The tricky question is how to commit to the value γ before seeing R and m? We overcome this problem by assuming that $\mathcal{A}$ executes the verification algorithm $\mathsf{Verify}_{\mathsf{Sc}}(\cdot)$ before outputting its forgery (Y, γ), as done in [NSW09]. This is w.l.o.g. because for every adversary that does not verify its forgery, we can build an adversary that has the same advantage but verifies its attempted forgery. This step guarantees that the elements $Y \in \mathbb{G}$ and $g_T^t \in \mathbb{G}_T$ appears as outputs of group oracles, with Y appearing before g_T^t. We bound the probability that Y appears later than g_T^t to be ϵ_{RPP}.

Hence $\mathcal{B}$ simply needs to guess a priori which group oracle query outputs g_T^t. During this step, $\mathcal{B}$ recovers the value of γ using the coefficients of the polynomials representing Y and g_T^t, as explained in (4.5) and proved in Lemma 4.1. Note that $\mathcal{B}$ also needs to guess a priori which element will be output as Y. $\mathcal{B}$ forwards the value of γ to the (now RPP) challenger $\mathcal{C}$ and obtains the random prefix R, which it uses as the encoding of g_T^t. Note that the probability that $\mathcal{B}$ will succeed in all the three guesses is at least $\frac{1}{2(3q)^2}$, where we later show that the number of elements to choose from is at most $3q$ in both the cases. We would like to note that recovering γ in the proof of [NSW09] (for the original Schnorr signature scheme) is easier than in the bilinear setting. This is because in [NSW09] it involved guessing an element in only one list and the polynomials involved are all binomials.

We now formally describe the game $\mathcal{G}$. The description of the group oracles is typical for proofs in the generic group model (see [Sho97, Mau05, BB08, GV12]).

Description of Game $\mathcal{G}$: Initially, $\mathcal{B}$ will choose a random bit $\beta_{\mathcal{C}} \xleftarrow{\$} \{0,1\}$. This bit decides which of the two problems RPSP (if $\beta_{\mathcal{C}} = 0$) or RPP (if $\beta_{\mathcal{C}} = 1$) will $\mathcal{B}$ attempt to solve using the forgery output by $\mathcal{A}$. If $\beta_{\mathcal{C}} = 0$, then $\mathcal{B}$ randomly chooses $i^* \xleftarrow{\$} \{1, \ldots, q\}$, else it randomly chooses $i^*, j^* \xleftarrow{\$} \{1, \ldots, 3q\}$. The quantity i^* indicates the step in which $\mathcal{B}$ interacts with $\mathcal{C}$ to obtain a random prefix $\xi_{T,i^*} \in \Xi_T$. This step may be a signature query (if $\beta_{\mathcal{C}} = 0$) or a group oracle query to $\mathcal{O}_T$ (if $\beta_{\mathcal{C}} = 1$). More on this will be discussed later when describing the simulation of signature queries and queries to the group oracle $\mathcal{O}_T$.

Let X, $\{\mathsf{T}_i : i \geq 1\}$, $\{\mathsf{U}_i : i \geq 1\}$ and $\{\mathsf{V}_i : i \geq 1\}$ be indeterminates, and $\{m_i : i \geq 1\}$ be bit-strings (messages) that are chosen by $\mathcal{A}$. Intuitively, these (or other) polynomials represent the relation amongst the group elements that are output by a group oracle, or guessed by $\mathcal{A}$. The indeterminate X corresponds to the quantity x (discrete logarithm of the secret key), whereas T_i corresponds to the parameter t_i chosen in the i^{th} signing step ($1 \leq i \leq q_\Omega$). Since $\mathcal{A}$ can query the group oracles with representations (from Ξ and Ξ_T) not previously obtained from the group oracles, in order to accommodate this case, we introduce the indeterminates U_i, V_i. The U_i correspond to the guessed elements of $\mathbb{G}$, whereas V_i correspond to the guessed elements of $\mathbb{G}_T$. We denote the lists $\{\mathsf{T}_i : i \geq 1\}$, $\{\mathsf{U}_i : i \geq 1\}$ and $\{\mathsf{V}_i : i \geq 1\}$ by $\{\mathsf{T}\}$, $\{\mathsf{U}\}$ and $\{\mathsf{V}\}$, respectively.

$\mathcal{B}$ maintains three lists of pairs

$$\mathcal{L} = \{(\mathsf{F}_{1,i}, \xi_{1,i}) : 1 \leq i \leq \tau_1\}, \tag{4.1}$$

$$\mathcal{L}_T = \{(\mathsf{F}_{T,i}, \xi_{T,i}) : 1 \leq i \leq \tau_T\}, \tag{4.2}$$

$$\mathcal{L}_\Omega = \{(m_i, \xi_{\Omega,i}, \gamma_i) : 1 \leq i \leq \tau_\Omega\}. \tag{4.3}$$

The entries $\mathsf{F}_{1,i} \in \mathbb{Z}_p[\mathsf{X}, \{\mathsf{U}\}, \{\mathsf{T}\}]$, $\mathsf{F}_{T,i} \in \mathbb{Z}_p[\mathsf{X}, \{\mathsf{U}\}, \{\mathsf{V}\}, \{\mathsf{T}\}]$ are multivariate polynomials over $\mathbb{Z}_p$, whereas $\xi_{1,i}$, $\xi_{\Omega,i}$, and $\xi_{T,i}$ are bit-strings in the encoding

sets Ξ (of $\mathbb{G}$), Ξ, and Ξ_T (of $\mathbb{G}_T$), respectively. We have $m_i \in \{0,1\}^*$ and $\gamma_i \in \mathbb{Z}_p$. The polynomials in lists $\mathcal{L}$ and $\mathcal{L}_T$ correspond to elements of $\mathbb{G}$ and $\mathbb{G}_T$, respectively, that $\mathcal{A}$ will ever be able to compute or guess. The list $\mathcal{L}_\Omega$ records the signatures obtained by $\mathcal{A}$ on the messages m_i of its choice. The values τ_1, τ_T and τ_Ω denote the respective list counters.

Initially, $\tau_1 = 1$, $\tau_T = 1$, $\tau_\Omega = 0$, $\mathcal{L} = \{ (1, \xi_{1,1}) \}$, $\mathcal{L}_T = \{ (\mathsf{X}, \xi_{T,1}) \}$, and $\mathcal{L}_\Omega = \{\}$. The bit-strings $\xi_{1,1}$, $\xi_{T,1}$ are set to random distinct strings from Ξ and Ξ_T, respectively. We assume that there is some ordering among the strings in the sets Ξ and Ξ_T (say, lexicographic ordering), so that given a string $\xi_{1,i}$ or $\xi_{T,i}$, it is possible to efficiently determine its index in the lists, if it exists. The initial state of the lists $\mathcal{L}$ and $\mathcal{L}_T$ correspond to the generator of $\mathbb{G}$ and the public key, respectively.

The game begins by $\mathcal{B}$ providing $\mathcal{A}$ with the string $\xi_{1,1}$ from $\mathcal{L}$, and the string $\xi_{T,1}$ from $\mathcal{L}_T$.

Signature Query: Signature queries by $\mathcal{A}$ are modeled as follows. $\mathcal{A}$ provides a message $m_{\tau_\Omega} \in \{0,1\}^*$ of its choice to $\mathcal{B}$. In response $\mathcal{B}$ first increments the counters $\tau_1 := \tau_1 + 1$, $\tau_T := \tau_T + 1$ and $\tau_\Omega := \tau_\Omega + 1$, and sets $\mathsf{F}_{T,\tau_T} := \mathsf{T}_{\tau_\Omega}$.

- **(RPSP Challenge)** If $\beta_{\mathcal{C}} = 0$ and $i^* = \tau_\Omega$, then $\mathcal{B}$ passes on m_{τ_Ω} to $\mathcal{C}$ to obtain a random prefix $\xi_{T,i^*} \in \Xi_T$ as part of an RPSP challenge. If ξ_{T,i^*} is already present in $\mathcal{L}_T$, then $\mathcal{B}$ completes the RPSP challenge with $\mathcal{C}$ by returning arbitrary values, after $\mathcal{A}$ terminates. Denote this event by Abort. Else $\mathcal{B}$ sets $\xi_{T,\tau_T} := \xi_{T,i^*}$.
- Else if $\beta_{\mathcal{C}} \neq 0$ or $i^* \neq \tau_\Omega$, then $\mathcal{B}$ sets ξ_{T,τ_T} to a random string distinct from those already present in $\mathcal{L}_T$.

Append $\mathcal{L}_T$ with $(\mathsf{F}_{T,\tau_T}, \xi_{T,\tau_T})$. $\mathcal{B}$ computes $\gamma_{\tau_\Omega} := \mathsf{H}(\xi_{T,\tau_T} || m_{\tau_\Omega})$, sets $\mathsf{F}_{1,\tau_1} := \mathsf{T}_{\tau_\Omega} + \gamma_{\tau_\Omega}\mathsf{X}$, sets ξ_{1,τ_1} to a random distinct string, appends $\mathcal{L}$ with $(\mathsf{F}_{1,\tau_1}, \xi_{1,\tau_1})$, sets $\xi_{\Omega,\tau_\Omega} := \xi_{1,\tau_1}$, and appends $\mathcal{L}_\Omega$ and provides $\mathcal{A}$ with $(m_{\tau_\Omega}\ ,\ \xi_{\Omega,\tau_\Omega}\ ,\ \gamma_{\tau_\Omega})$.

Group Operation of $\mathbb{G}$: The calls made by $\mathcal{A}$ to the group oracle $\mathcal{O}$ are modeled as follows. For group operations in $\mathbb{G}$, $\mathcal{A}$ provides $\mathcal{B}$ with two operands (bit-strings) $\xi_{1,i}, \xi_{1,j}$ $(1 \leq i, j \leq \tau_1)$ in $\mathcal{L}$ and also specifies whether to multiply or divide them. $\mathcal{B}$ answers the query by first incrementing the counter $\tau_1 := \tau_1 + 1$, and computes the polynomial $\mathsf{F}_{1,\tau_1} := F_{1,i} \pm F_{1,j}$. If $\mathsf{F}_{1,\tau_1} = \mathsf{F}_{1,k}$ for some $k < \tau_1$, then $\mathcal{B}$ sets $\xi_{1,\tau_1} := \xi_{1,k}$. Otherwise, ξ_{1,τ_1} is set to a random string distinct from those already present in $\mathcal{L}$. The pair $(\mathsf{F}_{1,\tau_1}, \xi_{1,\tau_1})$ is appended to $\mathcal{L}$ and $\mathcal{B}$ provides $\mathcal{A}$ with ξ_{1,τ_1}. Note that the (total) degree of the polynomials $\mathsf{F}_{1,i}$ in $\mathcal{L}$ is at most one.

If $\mathcal{A}$ queries $\mathcal{O}$ with an encoding ξ not previously output by the oracle, then $\mathcal{A}$ increments the counter $\tau_1 := \tau_1 + 1$, sets $\xi_{1,\tau_1} := \xi$, and sets $\mathsf{F}_{1,\tau_1} := \mathsf{U}_{\tau_1}$. The pair $(\mathsf{F}_{1,\tau_1}, \xi_{1,\tau_1})$ is appended to $\mathcal{L}$. This step is carried out for each guessed operand.

Group Operation of $\mathbb{G}_T$: The group oracle $\mathcal{O}_T$ is modeled similar to $\mathcal{O}$, instead appropriately updating the counter τ_T, and appending the list $\mathcal{L}_T$

with the output $(\mathsf{F}_{T,\tau_T}, \xi_{T,\tau_T})$. $\mathcal{B}$ provides $\mathcal{A}$ with ξ_{T,τ_T}. For guessed operands in $\mathbb{G}_T$, a new variable T_{τ_T} is introduced instead.

Pairing Operation: For a pairing operation, $\mathcal{A}$ queries $\mathcal{B}$ with two operands $\xi_{1,i}, \xi_{1,j}$ $(1 \leq i, j \leq \tau_1)$ in $\mathcal{L}$. $\mathcal{B}$ first increments $\tau_T := \tau_T + 1$, and then computes the polynomial $\mathsf{F}_{T,\tau_T} := \mathsf{F}_{1,i} \cdot \mathsf{F}_{1,j}$. Again, if $\mathsf{F}_{T,\tau_T} = \mathsf{F}_{T,k}$ for some $k < \tau_T$, then $\mathcal{B}$ sets $\xi_{T,\tau_T} := \xi_{T,k}$. Otherwise, ξ_{T,τ_T} is set to a random string distinct from those already present in $\mathcal{L}_T$. The pair $(\mathsf{F}_{T,\tau_T}, \xi_{T,\tau_T})$ is appended to $\mathcal{L}_T$, and $\mathcal{B}$ provides $\mathcal{A}$ with ξ_{T,τ_T}. Note that the degree of the polynomials $\mathsf{F}_{T,i}$ in $\mathcal{L}_T$ is at most two.

RPP Challenge: Recollect that $\mathcal{B}$ has earlier sampled $i^*, j^* \xleftarrow{\$} \{1, \ldots, 3q\}$. Since $\mathcal{A}$ makes at most $q_{\mathcal{O}} < q$ group oracle queries and that in each query $\mathcal{A}$ can guess at most two new elements, it is easy to see that lists $\mathcal{L}$ and $\mathcal{L}_T$ together have at most $3(q_{\mathcal{O}} + q_\Omega) \leq 3q$ elements. Hence

$$|\mathcal{L}| + |\mathcal{L}_T| \leq 3q. \tag{4.4}$$

If $\beta_{\mathcal{C}} = 1$, then during each of the queries above the counter τ_T is checked while adding an element to the list $\mathcal{L}_T$. If $i^* = \tau_T$, then $\mathcal{B}$ computes

$$\gamma^* = \sum_{i=1}^{q_\Omega} a_i \gamma_i \; - \; a_{\mathsf{X}}, \tag{4.5}$$

where a_{X} is the coefficient of X in F_{T,i^*}, a_i is the coefficient of T_i in F_{1,j^*} $(1 \leq i \leq q_\Omega)$, and γ_i is, as defined previously, the hash value in the i^{th} signature query.

If F_{1,j^*} does not exist, or $i^* > \tau_T$ at the end of the game $\mathcal{G}$, then $\mathcal{B}$ completes the RPP challenge with $\mathcal{C}$ by returning arbitrary values. Else, $\mathcal{B}$ passes $\gamma^* \in \mathbb{Z}_p$ to $\mathcal{C}$ to obtain a random prefix $\xi_{T,i^*} \in \Xi_T$, as part of an RPP challenge. If $\mathsf{F}_{T,\tau_T} = \mathsf{F}_{T,k}$ for some $k < \tau_T$ and $\xi_{T,i^*} \neq \xi_{T,k}$, then $\mathcal{B}$ completes the RPP challenge with $\mathcal{C}$ by returning arbitrary values (Abort). Else, if there is no such k but ξ_{T,i^*} is already present in $\mathcal{L}_T$, then also Abort. Else $\mathcal{B}$ sets $\xi_{T,\tau_T} := \xi_{T,i^*}$.

If $\mathcal{B}$ has made right guesses for i^* and j^*, then F_{1,j^*} and F_{T,i^*} corresponds to the forgery and satisfy

$$\mathsf{F}_{T,i^*} := \mathsf{F}_{1,j^*} - \gamma \mathsf{X}, \tag{4.6}$$

where γ is the hash value corresponding to the forgery. Note again that both the polynomials exist (in case of successful forgery) because we assume that $\mathcal{A}$ always verifies its attempted forgery before it is output. Lemma 4.1 below proves that indeed $\gamma^* = \gamma$. Because $\mathcal{A}$ has access to the oracle $\mathcal{O}_T$, it is easy to see that it is not possible to recover γ from F_{T,i^*} alone.

Lemma 4.1. *Let $\mathsf{F}_{T,i^*} = \mathsf{F}_{1,j^*} - \gamma\mathsf{X}$, as computed in (4.6). Let a_X be the coefficient of X in F_{T,i^*}, and a_i be the coefficient of T_i in F_{1,j^*} $(1 \leq i \leq q_\Omega)$. Also let γ_i be the hash value in the i^{th} signature query. Then $\gamma = \sum_{i=1}^{q_\Omega} a_i\gamma_i - a_{\mathsf{X}}$.*

Proof. Any polynomial in $\mathcal{L}$, in particular F_{1,j^*}, is of the form $\mathsf{F}_{1,j^*} = c_1 + \sum_{i=1} c_{2,i}\mathsf{U}_i + \sum_{i=1}^{q_\Omega} a_i(\mathsf{T}_i + \gamma_i\mathsf{X})$, where $c_1, c_{2,i}, a_i \in \mathbb{Z}_p$ are chosen by $\mathcal{A}$. Hence the lemma follows. □

End of Game $\mathcal{G}$: When $\mathcal{A}$ terminates it outputs $(m, (\xi_{1,\alpha}, \gamma)) \in \{0,1\}^* \times \Xi \times \mathbb{Z}_p$, where $\xi_{1,\alpha} \in \mathcal{L}$ and $1 \leq \alpha \leq \tau_1$. This corresponds to the "forgery" output by $\mathcal{A}$ in the actual interaction. $\mathcal{B}$ simply forwards m to its challenger $\mathcal{C}$.

Let Forge denote the event of successful forgery. Next, $\mathcal{B}$ chooses random values x, $\{u\}$, $\{v\}$, $\{t\} \leftarrow \mathbb{Z}_p$ for the indeterminates X, $\{\mathsf{U}\}$, $\{\mathsf{V}\}$, $\{\mathsf{T}\}$, respectively. Then it evaluates the polynomials in lists $\mathcal{L}$ and $\mathcal{L}_T$. $\mathcal{B}$ will abort if:

1. $\mathsf{F}_{1,i}(x, \{u\}, \{t\}) = \mathsf{F}_{1,j}(x, \{u\}, \{t\})$ in $\mathbb{Z}_p$, for any $\mathsf{F}_{1,i} \neq \mathsf{F}_{1,j}$ in $\mathcal{L}$.
2. $\mathsf{F}_{T,i}(x, \{u\}, \{v\}, \{t\}) = \mathsf{F}_{T,j}(x, \{u\}, \{v\}, \{t\})$ in $\mathbb{Z}_p$, for any $\mathsf{F}_{T,i} \neq \mathsf{F}_{T,j}$ in $\mathcal{L}_T$.

Let $\mathsf{Collide}$ denote either of the above events, i.e. a *collision* occurring in lists $\mathcal{L}$ and/or $\mathcal{L}_T$. This completes the description of game $\mathcal{G}$ and simulator $\mathcal{B}$.

Analysis of $\mathbf{Pr}^{forge}_{\mathcal{A},\Pi_{\mathsf{Sc}}}$: The success probability $\Pr^{forge}_{\mathcal{A},\Pi_{\mathsf{Sc}}}$ of $\mathcal{A}$ in the actual EUF-CMA game satisfies

$$\Pr^{forge}_{\mathcal{A},\Pi_{\mathsf{Sc}}} \leq \Pr[\mathsf{Forge} \mid \overline{\mathsf{Collide}}] + \Pr[\mathsf{Collide}]. \tag{4.7}$$

This is because the event $\overline{\mathsf{Collide}}$ ensures that $\mathcal{A}$ will get to see only distinct group elements in the actual interaction. In other words, $\mathcal{A}$ is unable to cause *collisions* among group elements. As long as the event $\mathsf{Collide}$ does not occur, then the view of $\mathcal{A}$ is identical in the game $\mathcal{G}$ and the actual interaction. Hence if $\mathcal{A}$ is unable to provoke collisions, then adaptive strategies are no more powerful than non-adaptive ones (see [Mau05, Lemma 2 on pp. 12], also [Sho97]). This observation allows us to choose group elements and their representations independently of the strategy of $\mathcal{A}$.

First we bound $\Pr[\mathsf{Collide}]$. The τ_1 polynomials $\mathsf{F}_{1,i}$ in $\mathcal{L}$ have degree at most one. Note that $\mathsf{F}_{1,i} \neq \mathsf{F}_{1,j} \Leftrightarrow \mathsf{F}_{1,i} - \mathsf{F}_{1,j} \neq 0$ as polynomials. From Lemma 2.3 (with $\lambda' = 0$), the probability that two distinct polynomials in $\mathcal{L}$ evaluate to the same value for randomly and independently chosen values for the indeterminates is at most $\frac{1}{p}$. Summing up over at most $\binom{\tau_1}{2}$ distinct pairs (i, j), the probability that the condition 1 above holds is at most $\binom{\tau_1}{2} \cdot \frac{2}{p}$. Similarly, the probability that the condition 2 above holds is at most $\binom{\tau_T}{2} \cdot \frac{2}{p}$. Using (4.4) we obtain

$$\Pr[\mathsf{Collide}] \leq \binom{\tau_1}{2} \cdot \frac{1}{p} + \binom{\tau_T}{2} \cdot \frac{2}{p} \leq \frac{1}{p}(\tau_1 + \tau_T)^2 \leq \frac{9q^2}{p}. \tag{4.8}$$

Next we bound $\Pr[\mathsf{Forge} \mid \overline{\mathsf{Collide}}]$ in terms of the advantage of $\mathcal{B}$ against $\mathcal{C}$. Whenever $\mathcal{A}$ succeeds in outputting a forgery $(m, (\xi_{1,\alpha}, \gamma))$, there are only two possibilities that can arise:

- **(Solving RPSP Challenge)** There exists an i $(1 \leq i \leq q_\Omega)$ such that $(m_i, (\xi_{1,\alpha}, \gamma)) \in \mathcal{L}_\Omega$. In other words, $\mathcal{A}$ uses a signature previously obtained to output its forgery on a distinct message. Let Forge_1 denote this event. If $\beta_\mathcal{C} = 0$ and $i^* = i$, then $\mathcal{B}$ can successfully use the forgery to solve the RPSP$[\Xi_T]$ problem for H. This is because $\mathcal{B}$ will attempt to solve the RPSP problem only when $\beta_\mathcal{C} = 0$, the probability of which is $\frac{1}{2}$. Since at the beginning itself $\mathcal{B}$ will decide at which signing step (step i^*) it will interact with $\mathcal{C}$ when $\beta_\mathcal{C} = 0$, the probability that $i^* = i$ is at least $\frac{1}{q_\Omega} > \frac{1}{q}$. Hence the advantage of $\mathcal{B}$ in solving RPSP problem is at least $\frac{1}{2q}\Pr[\mathsf{Forge}_1 \mid \overline{\mathsf{Collide}}] - \left(\frac{3q}{p}\right)$, where $\left(\frac{3q}{p}\right)$ is an upper bound on the probability that $\mathcal{B}$ does not Abort due to a repeated entry in $\mathcal{L}_T$ during RPSP challenge step. It may be noted that if $\mathcal{B}$ attempts to solve the RPP problem using this type of forgery, then Abort will occur with overwhelming probability. Therefore, $\Pr[\mathsf{Forge}_1 \mid \overline{\mathsf{Collide}}] \leq 2q \cdot \epsilon_{\mathrm{RPSP}} + \frac{6q^2}{p}$.
- **(Solving RPP Challenge)** The complementary event of Forge_1, $\overline{\mathsf{Forge}_1}$. That is, $\mathcal{A}$ does not use a signature previously obtained to output its forgery. Since $\mathcal{A}$ verifies its forgery before it is output, then there exists some i^{th} entry $(\mathsf{F}_{T,i}, \xi_{T,i})$ in the list $\mathcal{L}_T$ such that $\mathsf{H}(\xi_{T,i}||m) = \gamma$. Also let this entry be the first occurrence of this pair in $\mathcal{L}_T$. If $\beta_\mathcal{C} = 1$, $i^* = i$ and $j^* = \alpha$, then $\mathcal{B}$ can successfully use the forgery to solve the RPP$[\Xi_T]$ problem for H. Hence the advantage of $\mathcal{B}$ in solving the RPP problem is at least $\frac{1}{2(3q)^2}\Pr[\overline{\mathsf{Forge}_1} \mid \overline{\mathsf{Collide}}] - \left(\frac{3q}{p} + \epsilon_{\mathrm{RPP}}\right)$, where again $\left(\frac{3q}{p}\right)$ is an upper bound on the probability that $\mathcal{B}$ does not Abort due to a repeated entry in $\mathcal{L}_T$ during RPP challenge step.

  The quantity ϵ_{RPP} appearing above is an upper bound on the probability that the entry $(\mathsf{F}_{T,i}, \xi_{T,i})$ does not appear before $(\mathsf{F}_{1,\alpha}, \xi_{1,\alpha})$. Because $\mathsf{F}_{T,i} = \mathsf{F}_{1,\alpha} - \gamma\mathsf{X}$ (c.f. (4.6)) and that encodings are random, this means that $\mathcal{A}$ is able to compute the value γ even before getting an encoding $\xi_{T,i}$ such that $\mathsf{H}(\xi_{T,i}||m) = \gamma$. In other words, $\mathcal{A}$ has solved the RPP$[\Xi_T]$ problem for H. Therefore, $\Pr[\overline{\mathsf{Forge}_1} \mid \overline{\mathsf{Collide}}] \leq 36q^2 \cdot \epsilon_{\mathrm{RPP}} + \frac{108q^3}{p}$.

Since $\Pr[\mathsf{Forge} \mid \overline{\mathsf{Collide}}] = \Pr[\mathsf{Forge}_1 \mid \overline{\mathsf{Collide}}] + \Pr[\overline{\mathsf{Forge}_1} \mid \overline{\mathsf{Collide}}]$, we obtain

$$\Pr[\mathsf{Forge} \mid \overline{\mathsf{Collide}}] \leq 2q \cdot \epsilon_{\mathrm{RPSP}} + 36q^2 \cdot \epsilon_{\mathrm{RPP}} + \frac{6q^2}{p} + \frac{108q^3}{p}. \tag{4.9}$$

From (4.7), (4.8) and (4.9), we have $\mathrm{Pr}^{forge}_{\mathcal{A},\Pi_{\mathsf{Sc}}} \leq 2q \cdot \epsilon_{\mathrm{RPSP}} + 36q^2 \cdot \epsilon_{\mathrm{RPP}} + \frac{15q^2}{p} + \frac{108q^3}{p}$. Hence if $q = poly(\log p)$, then $\mathrm{Pr}^{forge}_{\mathcal{A},\Pi_{\mathsf{Sc}}}$ is negligible provided $(\epsilon_{\mathrm{RPSP}} + \epsilon_{\mathrm{RPP}})$ is negligible. This completes the proof of Theorem 4.1. □

4.3 A Leakage-Resilient Pairing-based Schnorr Signature Scheme

In this section, we describe a leakage-resilient variant Π^*_{Sc} of the scheme Π_{Sc}. We use the techniques of [KP10a] to transform Π_{Sc} to Π^*_{Sc}. A major difference between the two variants is that the secret key $X = g^x$ of Π_{Sc} is now split into two parts as $(S_0 := g^{l_0}, S'_0 := g^{x-l_0})$ for a random $l_0 \leftarrow \mathbb{Z}_p$. The two shares reside in different parts of the memory. The key generation step $\mathsf{KeyGen}^*_{\mathsf{Sc}}$ of Π^*_{Sc} is obtained by suitably modifying the $\mathsf{KeyGen}_{\mathsf{Sc}}$ step of Π_{Sc}. The signing step of Π^*_{Sc} is also split into two steps $\mathsf{Sign}^*_{\mathsf{Sc1}}$ and $\mathsf{Sign}^*_{\mathsf{Sc2}}$. After every signature query, the two shares of the secret key are randomly refreshed. This is required because, as seen in Section 3.2.1, if the secret state is not stateful, then the scheme cannot be secure in the presence of continual leakage.

Let $\mathsf{H} : \{0,1\}^* \rightarrow \mathbb{Z}_p$ be a hash function. The stateful signature scheme $\Pi^*_{\mathsf{Sc}} = (\mathsf{KeyGen}^*_{\mathsf{Sc}}, \mathsf{Sign}^*_{\mathsf{Sc1}}, \mathsf{Sign}^*_{\mathsf{Sc2}}, \mathsf{Verify}^*_{\mathsf{Sc}})$, defined on $\{0,1\}^*$, is as follows:

1. $\mathsf{KeyGen}^*_{\mathsf{Sc}}(\kappa)$: Compute $\mathbb{PP} := (\mathbb{G}, \mathbb{G}_T, p, e, g) \leftarrow \mathsf{BGen}(\kappa)$. Choose random $x, l_0 \leftarrow \mathbb{Z}_p$. Set $X := g^x$ and $X_T := e(g, X) = e(g, g)^x$. The public key is $pk := (\mathbb{PP}, X_T, \mathsf{H})$ and the secret key is $sk^* := (S_0 := g^{l_0}, S'_0 := g^{x-l_0} = X \cdot g^{-l_0}) \in \mathbb{G}^2$.
2. $\mathsf{Sign}^*_{\mathsf{Sc1}}(S_{i-1}, m_i)$: Choose random $t_i, l_i \leftarrow \mathbb{Z}_p$. Set $S_i := S_{i-1} \cdot g^{l_i}$, $\gamma_i := \mathsf{H}(g_T^{t_i} || m_i)$, and $Y'_i := g^{t_i} \cdot S_i^{\gamma_i}$.
3. $\mathsf{Sign}^*_{\mathsf{Sc2}}(S'_{i-1}, (Y'_i, \gamma_i, l_i))$: Set $S'_i := S'_{i-1} \cdot g^{-l_i}$, $Y_i := Y'_i \cdot (S'_i)^{\gamma_i}$, and $\sigma_i := (Y_i, \gamma_i)$. Output the signature σ_i. .
4. $\mathsf{Verify}^*_{\mathsf{Sc}}(pk, m, \sigma)$: Let $\sigma = (Y, \gamma) \in \mathbb{G} \times \mathbb{Z}_p$. Set $\rho := e(Y, g) \star (g_T^x)^{-\gamma}$. Output the bit $b = 1$ (*valid*) if $\mathsf{H}(\rho || m) = \gamma$. Otherwise output $b = 0$ (*invalid*).

The index i used above refers to the number of times the signing algorithm has been invoked. For $i \geq 1$, let $Z_i := \sum_{j=0}^{i} l_j$. The correctness property of Π^*_{Sc} follows from Π_{Sc} since $S_i \cdot S'_i = g^{Z_i} \cdot g^{x-Z_i} = X$. The leakage functions $f_i()$ and $h_i()$ that the adversary specifies to the signing oracle would take the form $f_i(S_{i-1}, (l_i, t_i))$ and $h_i(S'_{i-1}, (Y'_i, \gamma_i, l_i))$ (cf. (3.1) and (3.2)).

The signing step of Π^*_{Sc} requires totally six exponentiations - four for $\mathsf{Sign}^*_{\mathsf{Sc1}}$ and two for $\mathsf{Sign}^*_{\mathsf{Sc2}}$. This quantity can be reduced to five if g^{l_i} is also passed on from $\mathsf{Sign}^*_{\mathsf{Sc1}}$ to $\mathsf{Sign}^*_{\mathsf{Sc2}}$. Note that the $\mathsf{Sign}_{\mathsf{Sc}}$ step of Π_{Sc} requires only three exponentiations.

Since the input/output behaviour of Π^*_{Sc} and Π_{Sc} is identical, from Theorem 4.1 we obtain that Π^*_{Sc} is secure in the GBG model in a non-leakage setting.

Lemma 4.2. *The signature scheme* Π^*_{Sc} *is* EUF-CMA *secure in the generic bilinear group model if the* RPP$[\Xi_T]$ *and* RPSP$[\Xi_T]$ *problems are hard for* H.

The following theorem shows that Π^*_{Sc} is resilient to continual leakage in the GBG model if RPP[Ξ_T] and RPSP[Ξ_T] problems are hard for the hash function H, and $\lambda < \frac{\log p}{2} - \omega(\log\log p)$, where λ is the leakage parameter.

Theorem 4.2. *The signature scheme Π^*_{Sc} is secure with leakage w.r.t. Definition 3.2 in the generic bilinear group model relative to the hardness of RPP[Ξ_T] and RPSP[Ξ_T] problems for H. Let the RPP and RPSP problems be $(\Gamma, \epsilon_{\mathrm{RPP}})$ and $(\Gamma, \epsilon_{\mathrm{RPSP}})$-hard, respectively. Then the advantage of a (Γ-time, q-query) adversary who gets at most λ bits of leakage per each invocation of $\mathsf{Sign}^*_{\mathsf{Sc1}}$ or $\mathsf{Sign}^*_{\mathsf{Sc2}}$ is $O\left(q^2\,\epsilon_{\mathrm{RPP}} + q\,\epsilon_{\mathrm{RPSP}} + \frac{q^3}{p} + \frac{q^2}{p}2^{2\lambda}\right)$.*

Let us briefly sketch the main ideas of the proof. Working on the lines of (4.7), the advantage of $\mathcal{A}$ is bounded by its success probabilities conditioned on the event whether or not a collision has occurred in the lists consisting of elements of $\mathbb{G}$ and $\mathbb{G}_T$. It is important to note that the proofs for the non-leakage setting (i.e. proof of Theorem 4.1) and the leakage setting would be the same conditioned on the fact that a collision has not occurred. The reason is that in the event of no collision, the adversary must either solve the RPP or the RPSP problem for the hash function in order to output a forgery (let us recall that a solution to either the RPP or the RPSP problem implies a collision for the hash function). Hence leakage on the secret state will not be useful in this case. Hence the success probability of $\mathcal{A}$ against Π_{Sc} and Π^*_{Sc} is the same in the event of no collision (that includes the event of guessing the representations of group elements using partial information about them).

However the probability that a collision occurs in the leakage setting is increased by a factor of at most $2^{2\lambda}$. This is because when $\mathcal{A}$ has access to leakage output $f_i(S_{i-1}, (l_i, t_i))$ and $h_i(S'_{i-1}, (Y'_i, \gamma_i, l_i))$ during i^{th} signature query, then in adversary's view the parameters t_i, l_i $(i \geq 1)$ are no longer uniformly distributed even though they are still independent. Hence $\mathcal{A}$ can now cause collisions among polynomials (in Conditions 1-2 on page 70) with increased probability. Each value t_i can only be leaked by f_i, hence at most λ bits of t_i can be leaked. Since l_i appears in both $f_i()$ and $h_i()$, at most 2λ bits of l_i can be leaked.

The only useful information that the leakage functions can provide to $\mathcal{A}$ is about the secret key X and the values t_i. This is because the values l_i are independent of the signatures generated. However $\mathcal{A}$ can use the leakages of l_i to eventually leak X. If $\mathcal{A}$ is able to compute X, then it can trivially forge a signature on a distinct message. The event of no collision, and the fact that X is not a "linear combination" of the inputs to the leakage functions, guarantees that $\mathcal{A}$ is unable to compute X.

Proof. Let $\mathcal{A}$ be a (Γ-time, q-query) adversary that can break the security of Π^*_{Sc}. Hence $\mathcal{A}$ can make totally at most q group oracle, pairing oracle and signing oracle queries, and runs in time at most Γ. In the count of q, even group oracle queries by leakage functions f_i, h_i $(i \geq 1)$ specified by $\mathcal{A}$ are also

included. Let the adversary $\mathcal{A}$ play the game $\mathcal{G}'$ described below. This game is an extension of game $\mathcal{G}$ described in the proof of Theorem 4.1. To avoid repetition, we only describe here the extensions that are not part of game $\mathcal{G}$. Let $\{L\}$ denote the list of indeterminates $\{L_i : 1 \leq i \leq q_\Omega\}$ that correspond to the values l_i in Π^*_{Sc}.

Game $\mathcal{G}'$: For each leakage function $f_i(S_{i-1}, (l_i, t_i))$ and $h_i(S'_{i-1}, (Y'_i, \gamma_i, l_i))$, $\mathcal{A}$ maintains a pair of lists $\left(\mathcal{L}^{f_i}, \mathcal{L}^{f_i}_T\right)$ and $\left(\mathcal{L}^{h_i}, \mathcal{L}^{h_i}_T\right)$, respectively. These lists contain polynomial and bit-string pairs. The polynomials in $\mathcal{L}^{f_i}$ and $\mathcal{L}^{h_i}$ belong to $\mathbb{Z}_p[\mathsf{X}, \{\mathsf{U}\}, \{\mathsf{T}\}, \{\mathsf{L}\}]$, and the corresponding bit-strings are from the encoding set Ξ of group $\mathbb{G}$. The polynomials in $\mathcal{L}^{f_i}_T$ and $\mathcal{L}^{h_i}_T$ are in the ring $\mathbb{Z}_p[\mathsf{X}, \{\mathsf{U}\}, \{\mathsf{V}\}, \{\mathsf{T}\}, \{\mathsf{L}\}]$, and the corresponding bit-strings are from the encoding set Ξ_T of group $\mathbb{G}_T$. Intuitively, the polynomials in lists $\mathcal{L}^{f_i}$ and $\mathcal{L}^{h_i}$ correspond to the elements of group $\mathbb{G}$ that can be computed by f_i and h_i, respectively, whereas the lists $\mathcal{L}^{f_i}_T$ and $\mathcal{L}^{h_i}_T$ correspond to the elements of $\mathbb{G}_T$. Every polynomial in $\mathcal{L}^{f_i}$ is of the form $c_{1,i}\mathsf{L}_i + c_{2,i}\sum_{j=0}^{i-1}\mathsf{L}_j + c_{3,i}\mathsf{D}_i$, where $c_{1,i}$, $c_{2,i}$, $c_{3,i} \in \mathbb{Z}_p$ are chosen by $\mathcal{A}$ and $\mathsf{D}_i \in \mathbb{Z}_p[\mathsf{X}, \{\mathsf{U}\}, \{\mathsf{T}\}]$ is in $\mathcal{L}$ (cf. (4.1)). Every polynomial in $\mathcal{L}^{h_i}$ is of the form

$$d_{1,i}\mathsf{L}_i + d_{2,i}\left(\mathsf{X} - \sum_{j=0}^{i-1}\mathsf{L}_j\right) + d_{3,i}\left(\mathsf{T}_i + \gamma_i\left(\sum_{j=0}^{i}\mathsf{L}_j\right)\right) + d_{4,i}\mathsf{W}_i, \qquad (4.10)$$

where $d_{1,i}$, $d_{2,i}$, $d_{3,i}$, $d_{4,i} \in \mathbb{Z}_p$ are also chosen by $\mathcal{A}$ and $\mathsf{W}_i \in \mathbb{Z}_p[\mathsf{X}, \mathsf{X}_0, \mathsf{X}_1, \{\mathsf{U}\}, \{\mathsf{T}\}]$ is in the list $\mathcal{L}$. Note that the polynomials in lists $\mathcal{L}^{f_i}$ and $\mathcal{L}^{h_i}$ are of degree at most one, and that they do not contain the monomial X. The polynomials in lists $\mathcal{L}^{f_i}_T$ and $\mathcal{L}^{h_i}_T$ are of degree at most two.

The game $\mathcal{G}'$ proceeds exactly as game $\mathcal{G}$ except that $\mathcal{A}$ can also obtain leakage through functions f_i and h_i in the i^{th} signature query. In particular, when $\mathcal{A}$ terminates it outputs $(m, (\xi_{1,\alpha}, \gamma)) \in \{0,1\}^* \times \Xi \times \mathbb{Z}_p$, where $\xi_{1,\alpha} \in \mathcal{L}$ and $1 \leq \alpha \leq \tau_1$. Let us denote by Forge^* the event of successful forgery by $\mathcal{A}$. Let $\mathsf{Collide}^*$ denote the event of a collision occurring in lists $\mathcal{L}$, $\mathcal{L}_T$, $\mathcal{L}^{f_i}$, $\mathcal{L}^{h_i}$, $\mathcal{L}^{f_i}_T$, $\mathcal{L}^{h_i}_T$ $(1 \leq i \leq q_\Omega)$. The polynomials are now evaluated with values chosen from independent distributions with min-entropy $\log p - 2\lambda$, not necessarily from an uniform distribution. The exact distribution depends on the leakage functions chosen by $\mathcal{A}$. Since we are only interested to upper bound the collision probability, we can safely assume that the simulator chooses the right distribution. Note that even in the leakage setting, adaptive strategies are no more powerful than non-adaptive ones, as observed in [AM11, pp. 691]. This completes the description of the game $\mathcal{G}'$.

Let $\Pr^{forge}_{\mathcal{A},\Pi^*_{\mathsf{Sc}}}$ denote the advantage of $\mathcal{A}$ in computing a forgery against Π^*_{Sc}. On the lines of (4.7), we can write

$$\Pr^{forge}_{\mathcal{A},\Pi^*_{\mathsf{Sc}}} \leq \Pr[\mathsf{Forge}^* \mid \overline{\mathsf{Collide}^*}] + \Pr[\mathsf{Collide}^*]. \qquad (4.11)$$

As mentioned before, conditioned on the event $\overline{\mathsf{Collide}^*}$, the view of the adversary $\mathcal{A}$ will be same in both the games $\mathcal{G}'$ and $\mathcal{G}$. This is because in both the cases $\mathcal{A}$ will get to see only distinct group elements. Hence, from (4.9), we have

$$\Pr[\mathsf{Forge}^* \mid \overline{\mathsf{Collide}^*}] \;\leq\; O\left(q^2\,\epsilon_{\mathrm{RPP}} + q\,\epsilon_{\mathrm{RPSP}} + \frac{q^3}{p}\right). \tag{4.12}$$

Lemma 4.3. $\Pr[\mathsf{Collide}^*] \leq O\left(\frac{q^2}{p}2^{2\lambda}\right)$.

Proof. To compute the required probability, the polynomials in lists $\mathcal{L}$, $\mathcal{L}_T$, $\mathcal{L}^{f_i}$, $\mathcal{L}^{h_i}$, $\mathcal{L}_T^{f_i}$, $\mathcal{L}_T^{h_i}$ $(1 \leq i \leq q_\Omega)$ are evaluated by choosing values from $\mathbb{Z}_p$ according to (independent) distributions with min-entropy at least $\log p - 2\lambda$. This is because $\mathcal{A}$ can obtain at most 2λ bits of leakage about l_i $(i = 0, \ldots, q_\Omega)$, and at most λ bits of t_i $(i = 1, \ldots, q_\Omega)$. According to Lemma 2.1, the values l_i, t_i have min-entropy at least $\log p - 2\lambda$ in the view of $\mathcal{A}$. The total length of all the lists is at most $O(q_\Omega + q_\mathcal{O}) = O(q)$. Hence there can be at most $O(q^2)$ pairs of distinct polynomials (of degree at most two) evaluating to the same value. From Lemma 2.3 (with $\lambda' = 2\lambda$), we obtain $\Pr[\mathsf{Collide}^*] \leq O\left(\frac{q^2}{p}2^{2\lambda}\right)$. □

From (4.11), (4.12) and Lemma 4.3, we have $\mathrm{Pr}^{forge}_{\mathcal{A},\Pi^*_{\mathsf{Sc}}} \leq O\Big(q^2\,\epsilon_{\mathrm{RPP}} + q\,\epsilon_{\mathrm{RPSP}} + \frac{q^3}{p} + \frac{q^2}{p}2^{2\lambda}\Big)$. This completes the proof of Theorem 4.2. □

4.4 Conclusion and Future Directions

In this work we presented a pairing-based Schnorr signature scheme and quantified its security against independent and continual leakage (i.e, against split-state leakage model) in the generic bilinear group model. In particular, we showed that allowing λ bits of leakage at each of the two phases of every round in the proposed scheme can be compared to decreasing the security of the pairing-based Schnorr scheme (without leakage) by a factor of at most $2^{2\lambda}$ in our leakage model. It seems that our results readily extend to the min-entropy leakage model of Chapter 2. Signing takes at most 5 exponentiations in $\mathbb{G}$ plus 1 exponentiation in $\mathbb{G}_T$; verification takes 1 pairing plus 1 exponentiation in $\mathbb{G}_T$. A suitable bilinear pairing group to implement our modification of the Schnorr scheme is the pairing-friendly curve BN-128 studied by Scott in [Sco11].

It is interesting to compare the relative efficiency and strength of our scheme and the FKPR scheme by Faust et al. [FKPR10]. The latter has a weak form of EUF-CMA security against continual independent leakages in the random oracle model, where the adversary can ask at most for D signatures queries, for D fixed before the key generation phase. The main advantage

of that construction with respect to ours is that it can be implemented over any group $\mathbf{G}$ where the DL problem is conjectured to be hard (our scheme needs pairing-based groups). Let us now examine its disadvantages against our scheme, which are all related to its practicality. The signer in the FKPR scheme needs to maintain a state consisting on roughly d Schnorr signatures and d public and corresponding secret keys, with the length of a signature being proportional to d and $D = 2^{d+1} - 2$; signing takes 9 exponentiations in the group $\mathbf{G}$, while verification time is proportional to d. FPKR only tolerates a leakage rate of roughly $1/36$. Thus, for reasonable values of d, e.g. $d = 20$, our scheme is more efficient in storage, computing time and leakage ratio than the FPKR scheme, while offering standard existential unforgeability against continual leakage in the split-state model. Finally both our scheme and the FPKR scheme use an idealized model of computation to prove security, namely the former uses the random oracle model, while ours uses generic groups.

One possible research direction is to see if the use of Forking lemma could eventually lead to a simplification of our leakage-resilience proof of the pairing-based Schnorr signature scheme, this time with respect to the hardness of the pairing inversion problem[1]. Although we have not really explored this possibility, we are rather sceptical about its success. This is due to the fact that in the GBG model we can assume knowledge of the secret key g^x to simulate the leakage. This is not the case in the Forking lemma case, where we would use an adversary to compute g^x from $e(g, g)^x$, i.e. we do not know how to simulate leakage on g^x, since we do not know g^x.

[1] We thank an anonymous referee of IMA CC 2013 for this suggestion.

Part II

Probing Leakage Model: Efficient Higher-Order Masking of S-boxes

Chapter 5

Masking of S-boxes and Polynomial Evaluation

In this chapter, we improve the efficiency of a generic higher-order masking scheme proposed by Carlet et al. [CGP+12]. Efficiency of this scheme is related to the problem of evaluating polynomials over binary finite fields representing S-boxes. The corresponding evaluation cost model considers only *non-linear multiplications*. A non-linear multiplication is a multiplication of two distinct non-constant polynomials over a binary field.

Firstly, we make a formal approach to this polynomial evaluation problem. To this end, we investigate optimal methods for exponentiation in $\mathbb{F}_{2^n}$ by studying a previously proposed variant of addition chain that we call as *cyclotomic-class addition chain*, or *CC-addition chain*. Among several interesting properties, we prove lower bounds on min-length CC-addition chains. We define the notion of $\mathbb{F}_{2^n}$-polynomial chain, and use it to count the number of non-linear multiplications required while evaluating polynomials over $\mathbb{F}_{2^n}$. We give lower bounds on the length of such a chain for any polynomial. In particular, we prove a lower bound of $\Omega(2^{n/2}/\sqrt{n})$ on the complexity of any method to evaluate polynomials over $\mathbb{F}_{2^n}$. As a consequence, we show that a lower bound for the masking complexity of DES S-boxes is 3 (upon padding outputs with zeroes), and that of PRESENT S-box is 2.

Next, we describe a new technique for evaluating polynomials over binary finite fields. For n-bit S-boxes our new technique has heuristic complexity $\mathcal{O}(2^{n/2}/\sqrt{n})$ instead of $\mathcal{O}(2^{n/2})$ proven complexity for the parity-split method. Hence our method is asymptotically optimal. In practice we can evaluate any 8-bit S-box in 10 non-linear multiplications instead of 22 needed by the parity-split method, and the DES S-boxes in 4 non-linear multiplications instead of 10. We also evaluate any 4-bit S-box in 2 non-linear multiplications instead of 3. Hence our method achieves optimal complexity for the PRESENT S-box.

Contents

5.1 Introduction

5.1.1 Masking

A well known technique to protect implementations against power analysis based side-channel attacks is to mask internal *secret* variables. This is done by XORing any internal variable with a random variable r, for e.g., $x' = x \oplus r$. However, this will make the implementation secure against first-order attacks only. Second-order attacks against such counter-measures is proposed in [Mes00]. In this type of attack the adversary combines the information obtained from two internal variables. This will require more data (power consumption traces) in practice, which could make the attack infeasible in

certain cases. In general the above masking technique can be extended to secure an implementation against higher-order attacks. This can be achieved by splitting an internal variable x into d shares, say, $x = \bigoplus_{i=1}^{d} x_i$. Using this idea it is easy to compute any linear/affine function ℓ in a secured way, since it is enough to compute $y_i = \ell(x_i)$ for $1 \leq i \leq d$. However, it is not obvious how to do this for non-linear functions. In practice, nearly every cryptographic primitive includes some non-linear function, e.g., S-box, modular addition, etc.

5.1.2 Generic Higher-Order Masking

The Rivain-Prouff masking scheme is the first provably secure higher-order masking technique for AES [RP10]. The main idea of this method is to perform secure monomial evaluation with d shares of a secret variable using the previously known ISW scheme [ISW03]. Namely, the non-linear part of the AES S-box can be represented by the monomial x^{254} over $\mathbb{F}_{2^8}$. Prouff and Rivain showed that this monomial can be evaluated securely using 4 *non-linear multiplications* and a few linear squarings. By using this scheme the AES S-box can be masked for any order d.

This method was extended to a generic technique for higher-order masking, in [CGP+12], by Carlet, Goubin, Prouff, Quisquater and Rivain (CGPQR). Any given n-bit S-box can be represented by a polynomial $\sum_{i=0}^{2^n-1} a_i\, x^i$ over $\mathbb{F}_{2^n}$ using Lagrange's interpolation theorem. Hence, any S-box can be masked by secure evaluation of this polynomial with d shares of a secret variable. This is the first generic technique to mask any S-box for any order d. In this technique a polynomial evaluation in $\mathbb{F}_{2^n}$ is split into simple operations over $\mathbb{F}_{2^n}$: addition, multiplication by constant, and regular multiplication of two elements. Note that multiplication of two same elements (i.e. squaring) and multiplication by a constant – both are linear operations over $\mathbb{F}_{2^n}$, hence easy to mask. For performing a secure multiplication of two distinct elements, *i.e.* a non-linear multiplication, the CGPQR masking scheme uses the ISW method as in [RP10].

Asymptotically, the running time of the Rivain-Prouff and CGPQR masking schemes is dominated by the number of non-linear multiplications required to evaluate a polynomial over $\mathbb{F}_{2^n}$. Namely with d shares, using the ISW method an affine function can be masked with only $\mathcal{O}(d)$ operations over $\mathbb{F}_{2^n}$, whereas a non-linear multiplication requires $\mathcal{O}(d^2)$ operations. Note that for achieving d-th order security the Rivain-Prouff scheme requires at least $2d + 1$ shares. Originally it was claimed in [RP10] that the scheme is secure against d-th order attack with $d+1$ shares. However, an attack of order $d/2$ was shown in [CPRR13] against the scheme. The authors of [CPRR13] also showed a d-th order secure scheme with $d + 1$ shares for some subset of S-boxes.

Another higher-order masking scheme has been proposed by Coron [Cor14]. This scheme is based on the table-recomputation technique [CJRR99, SP06,

RDP08]. An advantage of CGPQR scheme over Coron's scheme is that the former can be implemented with very little memory compared to the latter.

5.1.3 Masking and Polynomial Evaluation

An (n,m)-S-box is a function from $\{0,1\}^n$ to $\{0,1\}^m$, where $m \leq n$. For most of the well-known ciphers, n is 4, 6 or 8. To design a generic masking scheme, Carlet et al. [CGP$^+$12] consider a polynomial representation of an (n, m)-S-box over $\mathbb{F}_{2^n}$. The n-bit and m-bit strings are identified with elements of $\mathbb{F}_{2^n}$ in a natural way, if necessary, by appending m-bit strings with leading zeros. Such a polynomial can be easily computed from the S-box table by applying Lagrange interpolation method. The polynomial will be of the form $\sum_{i=0}^{2^n-1} a_i\, x^i$, where $a_i \in \mathbb{F}_{2^n}$. Hence the evaluation of an S-box reduces to evaluating the corresponding polynomial for some element in $\mathbb{F}_{2^n}$.

The operations involved in the above polynomial evaluation are: addition, multiplication by a scalar (from $\mathbb{F}_{2^n}$), squaring, and multiplications that are not squaring. Except the last one, all the above operations are affine in $\mathbb{F}_{2^n}$. In this masking scheme only the non-linear multiplications are significant. Because the dth-order masking of an affine operation requires $O(d)$ logical operations, whereas a non-linear multiplication requires $O(d^2)$ operations [CGP$^+$12]. Hence the *masking complexity* of a S-box is defined as the minimum number of non-linear multiplications needed to evaluate its corresponding polynomial.

Efficient methods for polynomial evaluation is a well-studied area [Knu97, Section 4.6.4]. Of particular interest is the evaluation of a power function (i.e. x^α), because of its simplicity. Not only are these functions of theoretical interest, there are also studies on the suitability of S-boxes based on power functions [NGG09]. Formal analysis of the optimal methods to evaluate these powers has led to a detailed study of *addition chains* [vzGN97, Knu97, Section 4.6.3]. The length of these chains correspond to the number of multiplications needed for the corresponding exponentiation. However, to analyze the number of non-linear multiplications required to evaluate an S-box, we need to investigate a variant of addition chain introduced in [CGP$^+$12]. We call this variant as *cyclotomic-class addition chain*, or in short, *CC-addition chain* to distinguish it from the usual addition chain. Also, CC-addition chains more accurately model the cost of exponentiations in $\mathbb{F}_{2^n}$. This is because squaring is very efficient in $\mathbb{F}_{2^n}$, and we can also use the relation $x^{2^n} = x$ to our advantage.

5.1.4 Our Contribution

In this chapter, we analyze and improve the generic higher order masking scheme proposed in [CGP$^+$12]. In Section 5.2, we start by establishing several interesting properties of CC-addition chain. We prove a lower bound on

the min-length CC-addition chain of any integer, which turns out to be logarithmic in the Hamming weight of the integer. As a consequence, we disprove a previous claim that integers of the form $2^n - 2$ have the longest min-length CC-addition chain than any other smaller number. We give a short mathematical proof showing that the masking complexity of AES is at least four, which was previously established by the brute-force method in [CGP+12]. We also give a result on the monotonicity property of the min-length CC-additions of an integer.

Next in Section 5.3, we propose and define the notion of $\mathbb{F}_{2^n}$-polynomial chain. Although the notion of CC-addition chain helps to evaluate the masking complexity of power functions (i.e., monomials), in case of general polynomials the idea of $\mathbb{F}_{2^n}$-polynomial is more natural and useful. Such a notion is necessary to formally define and establish lower bounds on the masking complexity of an S-box. We prove the well-definedness of the notion of masking complexity by arguing that it is invariant of the way of representing the corresponding field.

We prove two lower bounds on the minimum number of non-linear multiplications required to evaluate a polynomial in $\mathbb{F}_{2^n}$. This lower bound is related to the min-length CC-addition chains of the integers present in the exponents of the polynomial. As a corollary, we show that the masking complexity of DES (S-box) is at least three upon padding each 4-bit output with 2 leading zeroes. (Zero was a natural choice though the effect on complexity is unclear yet.) For PRESENT we show that its complexity is at least two. Previously, no such lower bounds were known. Next, we significantly improve this lower bound to $\Omega(2^{n/2}/\sqrt{n})$ non-linear multiplications for any method to evaluate arbitrary polynomials over $\mathbb{F}_{2^n}$.

Later in the section, we propose an improved generic technique for fast polynomial evaluation in $\mathbb{F}_{2^n}$. For arbitrary n-bit S-box, our method has heuristic complexity $\mathcal{O}(2^{n/2}/\sqrt{n})$, compared to the $\mathcal{O}(2^{n/2})$ proven complexity for the Parity-Split method from [CGP+12]. Hence our method is asymptotically optimal.

As a concrete application of our new polynomial evaluation method, we show in Section 5.4 that for the generic higher-order masking of several well known S-boxes, e.g. DES, CLEFIA, PRESENT, etc., our method reduces the number of multiplications compared to the previously known methods [CGP+12]. In particular, using our method PRESENT can be masked with 2 multiplications (instead of 3), and DES with 4 multiplications (instead of 10), see Table 5.1. In Table 5.8, we report the timing results for DES masked using our technique.

5.2 Evaluation of Powers

In this section, we consider optimal methods (that minimize non-linear multiplications) for evaluating monomials in $\mathbb{F}_{2^n}$.

Methods	S-box				
	DES	PRESENT	SERPENT	CAMELLIA	CLEFIA
Previous Work [CGP+12]	10	3	3	22	22
Our Method (Sec. 5.4)	**4**	**2**	**2**	**10**	**10**

Table 5.1: Number of non-linear multiplications required for the CGPQR generic higher-order masking scheme.

5.2.1 Definitions

Notation. $\nu(n)$ refers to the number of bits that are one in the binary representation of n, i.e. the Hamming weight of n. For a binary string z in $\{0,1\}^*$, $\langle z \rangle_2$ denotes the binary representation of some non-negative integer.

For any subset $\Lambda \subseteq \{0, 1, \ldots, 2^n - 2\}$, x^Λ denotes the set of monomials $x^\Lambda = \left\{x^i : i \in \Lambda\right\} \subseteq \mathbb{F}_{2^n}[x]$. Finally we denote by $\mathcal{P}(x^\Lambda)$ the set of all polynomials in $\mathbb{F}_{2^n}[x]$ whose monomials are only from the set x^Λ.

Let us recollect the standard notion of *addition chain*.

Definition 5.1. [Addition Chain [Knu97, Section 4.6.3]] *An* addition chain *S for α ($\alpha \in \mathbb{N}$) is a sequence of integers*

$$a_0 = 1,\ a_1,\ a_2,\ \ldots\ ,\ a_r = \alpha, \tag{5.1}$$

such that for every $i = 1, 2, \ldots, r$, there exist some $0 \leq j, k < i$ such that

$$a_i = a_j + a_k.$$

The length of S, denoted by $L(S)$, is r.

Thus in an addition chain, any element in the sequence (except the first) must be a sum of some previous two elements. The length of a shortest addition chain for α is denoted by $l(\alpha)$. Formally,

$$l(\alpha) = \min\left\{L(S) : S \text{ is an addition chain for } \alpha\right\}. \tag{5.2}$$

Intuitively, $l(\alpha)$ represents the minimum number of "multiplications" needed to compute x^α from x (x is an element of a *monoid*).

The notion of "addition chain" has been generalized to *q-addition chain* ($q \in \mathbb{N}$) in [vzG91]. In this generalization of the "usual" addition chains the multiple of an element by q can be computed in a single step. Note that an (usual) addition chain is a 2-addition chain.

The q-addition chains are more relevant than (2-)addition chains in the case of exponentiations in finite fields $\mathbb{F}_{q^n}$ of characteristic $q \neq 2$. In such a field it is possible to compute x^q very efficiently, often "free" [vzG91].

In this work we study another variant of addition chain introduced in [CGP+12]. Before we describe the variant, let us first see the following definition.

Definition 5.2. [Cyclotomic Class [CGP+12]] *Let $n \in \mathbb{N}$ and $\alpha \in \{0, 1, \ldots, 2^n - 2\}$. The* cyclotomic class *of α (w.r.t. n), denoted by C_α, is defined as*

$$C_\alpha = \left\{ \alpha \cdot 2^i \pmod{2^n - 1} \ : \ i = 0, 1, \ldots, n-1 \right\}.$$

The intuition for introducing the above definition comes from the following scenario. Let g be a generator of the multiplicative group $\mathbb{F}_{2^n}^\times$. Given $x = g^\alpha$, the set $\{x, x^2, x^4, x^8, \ldots, \}$ is the same as $\{g^i \mid i \in C_\alpha\}$. Note that $x^{2^n} = x$ in $\mathbb{F}_{2^n}^\times$. Since $2^n \equiv 1 \pmod{2^n - 1}$, therefore $|C_\alpha| \leq n$. It is easy to see that the relation R on set $\{0, 1, \ldots, 2^n - 2\}$, defined as $(\alpha, \beta) \in R$ iff $\beta \in C_\alpha$, is an equivalence relation. Hence the collection of cyclotomic classes forms a partition of the set $\{0, 1, \ldots, 2^n - 2\}$. Since $|C_\alpha| \leq n$, we obtain the following observation.

Remark 5.1 The number of cyclotomic classes w.r.t. n is at least $\frac{2^n-1}{n}$.

In [CGP+12], the exact count of the number of cyclotomic classes (w.r.t. n) is given as $\sum_{\delta \mid (2^n-1)} \frac{\phi(\delta)}{\mu(\delta)}$, where ϕ is the Euler's totient function and $\mu(\delta)$ is the multiplicative order of 2 modulo δ. However, no lower bound on this expression was given there. The simple observation in Remark 5.1 shows that $\sum_{\delta \mid (2^n-1)} \frac{\phi(\delta)}{\mu(\delta)} \geq \frac{2^n-1}{n}$.

A variant of addition chain proposed in [CGP+12] is the *cyclotomic-class addition chain*, in short, *CC-addition chain.*

Definition 5.3. [CC-Addition Chain [CGP+12]] *Let $n \in \mathbb{N}$, $\alpha \in \{1, 2, \ldots, 2^n - 2\}$, and $C = \{C_i : i = 0, 1, \ \ldots, 2^n - 2\}$ be the collection of cyclotomic classes w.r.t. n, A* cyclotomic-class addition chain *S_C of α (w.r.t. n) is a sequence of cyclotomic classes*

$$C_{a_0} = C_1, \ C_{a_1}, \ C_{a_2}, \ \ldots, \ C_{a_r} = C_\alpha, \tag{5.3}$$

such that for every $i = 1, 2, \ldots, r$, there exist some $0 \leq j, k < i$, $\beta_i \in C_{a_i}$, $\beta_j \in C_{a_j}$, and $\beta_k \in C_{a_k}$ such that

$$\beta_i \equiv \beta_j + \beta_k \pmod{2^n - 1}.$$

The length of S_C, denoted by $LC_n(S_C)$, is r.

Formally, a shortest CC-addition chain for α (w.r.t. n), denoted by $m_n(\alpha)$, is defined as

$$m_n(\alpha) = \min \{LC_n(S_C) : S_C \text{ is an addition chain for } \alpha \text{ (w.r.t. } n)\}. \quad (5.4)$$

The phrase "masking complexity of α" has been used in [CGP+12] to describe $m_n(\alpha)$. CC-addition chains describe a way to compute x^α from $x \in \mathbb{F}_{2^n}^\times$, where squaring operations are considered free and hence not counted. These sort of chains model the complexity of exponentiation in $\mathbb{F}_{2^n}$ more accurately than (2-)addition chains when squaring is implemented very efficiently using a special representation of field elements [vzG91]. CC-addition chains also model exactly the number of *non-linear* multiplications required to mask S-boxes that are represented by *power functions* [CGP+12]. An important difference between q-addition chains, in particular 2-addition chains, and CC-addition chains is that the former is a sequence of positive integers while the latter is a sequence of classes. It is for this reason that we refer to the latter chain as "cyclotomic-class addition chain" and not just 2-addition chain as done in [CGP+12]. The notion of CC-addition chains can be extended in a natural way to $\mathbb{F}_{q^n}$ to obtain q-CC-addition chain, analogous to q-addition chain. Accordingly, the CC-addition chain in Definition 5.3 may also be referred to as 2-CC-addition chain. In this work, we restrict ourselves to (2-)CC-addition chains, particularly keeping applications to higher-order masking in mind.

Note that $m_n(\alpha)$ is not necessarily equal to the minimum number of non-doubling steps in all of addition chains for α, though $m_n(\alpha) \leq l(\alpha)$. That is, every CC-addition chain does not necessarily need to be derived from an addition chain by not explicitly writing the doubling steps. This is a consequence of the fact that there exist α, n_1 and n_2 such that $m_{n_1}(\alpha) \neq m_{n_2}(\alpha)$. For example, $m_5(23) = 2$ but $m_6(23) = 3$. We refer to the table of values for $m_n(\alpha)$ for $n \leq 11$ in [CGP+12].

5.2.2 CC-Addition Chain: Upper Bound

Nevertheless, we can obtain upper bounds on the value of $m_n(\alpha)$ using previous results on addition chains in a straightforward way. Note that for a given value of α, $m_n(\alpha)$ is defined only for those n such that $\alpha \leq 2^n - 2$. Hence we require $n \geq \lceil \log_2(\alpha + 2) \rceil$.

A trivial upper bound $m_n(\alpha) \leq \nu(\alpha) - 1$ is obtained from the *binary method* [Knu97, Section 4.6.3]. Let $\alpha = b_t 2^t + b_{t-1} 2^{t-1} + \ldots + b_1 2^1 + b_0$, where $t = \lfloor \log_2 \alpha \rfloor$, $b_i \in \{0, 1\}$ $\forall i = 1, \ldots, t$, and $b_t = 1$. An addition chain obtained from the binary method is as follows

$$b_t = 1,\ b_t 2,\ b_t 2 + b_{t-1},\ 2(b_t 2 + b_{t-1}),\ b_t 2^2 + b_{t-1} 2 + b_{t-2},\ \ldots,\ \alpha.$$

The above addition chain yields a CC-addition chain for α (w.r.t. any $n \geq \lceil \log_2(\alpha + 2) \rceil$). Hence the length of such a chain is $\nu(\alpha) - 1$. Note that we count only those additions that are not doublings.

An improved upper bound for $m_n(\alpha)$ is possible if we use the techniques of Brauer [Bra39]. In [Bra39], addition chains much shorter than those from the binary method have been constructed. This result on (2-)addition chains has also been extended to q-addition chains in [vzG91]. See also [vzGN04, Knu97, Section 4.6.3].

Brauer's method of constructing addition chains is a generalization of the binary method mentioned above. Instead of working in the base-2 expansion of α, we now work with base-2^k expansion ($k \in \mathbb{N}$). Let $z = 2^k$ and $\alpha = b_t z^t + b_{t-1} z^{t-1} + \ldots + b_1 z^1 + b_0$, where $t = \lfloor \log_z \alpha \rfloor$, $b_i \in \{0, 1, \ldots, z-1\}$ $\forall i = 0, 1, \ldots, t$, and $b_t \neq 0$. The corresponding addition chain is

$$
\begin{aligned}
&1,\ 2,\ \ldots\ ,\ z-2,\ z-1,\\
&\quad b_t 2,\ b_t 4,\ \ldots\ ,\ b_t z,\ b_t z + b_{t-1},\\
&\qquad (b_t z + b_{t-1})\, 2,\ (b_t z + b_{t-1})\, 4,\ \ldots\ ,\ (b_t z + b_{t-1})\, z,\ b_t z^2 + b_{t-1} z + b_{t-2},\\
&\qquad\qquad \ldots \qquad b z^t + b_{t-1} z^{t-1} + \ldots + b_1 z^1 + z_0.
\end{aligned}
$$

The total length of the above addition chain is $z - 2 + t(k+1)$. The number of non-doubling steps is $(z-2)/2 + t = 2^{k-1} - 1 + \left\lfloor \frac{\log_2 \alpha}{k} \right\rfloor$, which is also the length of the corresponding CC-addition chain for α (w.r.t. any n). This value is minimized when $k \approx \log_2 \log_2 \alpha - 2 \log_2 \log_2 \log_2 \alpha$ and the corresponding value is about $\frac{\log_2 \alpha}{\log_2 \log_2 \alpha - 2 \log_2 \log_2 \log_2 \alpha} + \frac{\log_2 \alpha}{2(\log_2 \log_2 \alpha)^2} - 1$. Hence as $\alpha \to \infty$, we obtain

$$m_n(\alpha) \leq \frac{\log_2 \alpha}{\log_2 \log_2 \alpha} \left(1 + o(1)\right). \tag{5.5}$$

5.2.3 CC-Addition Chain: Lower Bound

No non-trivial lower bounds were known for $m_n(\alpha)$ prior to [RV13]. In this chapter we show that $m_n(\alpha) \geq \lceil \log_2(\nu(\alpha)) \rceil$. Recall that $\nu(\alpha)$ is the Hamming weight of α in the binary notation. The basic idea is to first show that Hamming weight is invariant in a cyclotomic class. To obtain the bound, we then use this result along with the simple fact that when two positive integers are added, then the Hamming weight of sum is at most the sum of the Hamming weights. Similar techniques have been used in [vzG91].

Lemma 5.1. *Let $n \in \mathbb{N}$, $\alpha \in \{0, 1, \ldots, 2^n - 2\}$, and C_α be the cyclotomic class of α (w.r.t. n). If $\beta \in C_\alpha$, then $\nu(\beta) = \nu(\alpha)$.*

Proof. This follows from a well-known observation that the multiplication of α by 2 modulo $2^n - 1$ is same as the cyclic left shift of the n-bit binary representation of α.

□

As an illustration, consider the cyclotomic class C_3 of $\alpha = 3$ w.r.t. $n = 5$. $C_3 = \{3, 6, 12, 24, 17\}$. Note that $17 \cdot 2 \equiv 3 \pmod{31}$. In the binary representation,

$$C_3 = \{\langle 00011\rangle_2, \langle 00110\rangle_2, \langle 01100\rangle_2, \langle 11000\rangle_2, \langle 10001\rangle_2\}. \tag{5.6}$$

The following proposition gives a lower bound for $m_n(\alpha)$.

Proposition 5.1. $m_n(\alpha) \geq \lceil \log_2(\nu(\alpha)) \rceil$.

Proof. From Lemma 5.1 and, the fact that the Hamming weight of sum of two positive integers is at most the sum of the Hamming weights, we obtain that the CC-addition chain of length at most r (5.3) can only contain integers having Hamming weight at most 2^r. This is because elements of C_1 have Hamming weight 1 and at each step the Hamming weight can at most double. Therefore, in order for α to be present in a CC-addition chain, then the chain's length must be at least $\lceil \log_2(\nu(\alpha)) \rceil$. □

As a consequence of the above proposition, we now disprove the claim made in [CGP+12, pp. 373]. Their claim was that given a (fixed) value of n, $m_n(2^n-2) \geq m_n(\alpha)\ \forall \alpha = 1, \ldots, 2^n-3$, i.e., 2^n-2 has the longest min-length CC-addition chain among the integers modulo $2^n - 1$.

Proposition 5.2. *Let $n = 2^t+1$ for some $t \in \mathbb{N}$ and $t > 2$. Then $m_n(2^n-2) = t$. In particular, $m_9(510) = 3 < m_9(508) = 4$.*

Proof. The proof proceeds in two steps. In lemma 5.2 below, we first show that $m_n(2^n - 2) = t$. As a result, $m_9(510) = 3$. Then in Lemma 5.3, we prove that $m_9(508) = 4$. This will complete the proof of the proposition. □

Lemma 5.2. *$m_n(2^n - 2) = t$, where $n = 2^t + 1$, $t \in \mathbb{N}$ and $t > 2$.*

Proof. From Proposition 5.1, we have $m_n(2^n - 2) \geq \log_2(\nu(2^n - 2)) = t$. A CC-addition chain of length t for $2^n - 2$ (w.r.t. n) can be constructed as follows

$$C_1,\ C_{2^2-1},\ C_{2^4-1},\ C_{2^8-1},\ \ldots\ ,\ C_{2^{2^t}-1} = C_{2^n-2}. \tag{5.7}$$

Note that $C_{2^{2^t}-1} = C_{2^n-2}$ because $2^n - 2 = 2\left(2^{2^t} - 1\right)$. Why the above sequence is indeed a CC-addition chain can be readily seen if we look at the n-bit-representations of the representatives of the cyclotomic classes in the above sequence. In the proof of Proposition 5.1 and the example in (5.6), we have observed that all the elements of a given cyclotomic class can be obtained by (left) cyclic shifts of the n-bit-representation of any one element of the class. Consider an integer sequence

$$\begin{aligned}&\langle 1\rangle_2 \overset{\times}{\to} \langle 10\rangle_2 \overset{+}{\to} \langle 11\rangle_2 \overset{\times}{\to} \langle 1100\rangle_2 \overset{+}{\to} \langle 1111\rangle_2 \to \\ &\ldots \to \langle \underbrace{11\ldots11}_{2^t}\rangle_2 \overset{\times}{\to} \langle \underbrace{11\ldots110}_{2^t}\rangle_2.\end{aligned} \tag{5.8}$$

In the above sequence, those arrows marked with $\times$ correspond to multiplying by a power of 2 (i.e. left shift) and hence such a step is not a separate step in the corresponding CC-addition chain. But those marked with + correspond to addition of two distinct integers and hence count as one step in the CC-addition chain. This shows that the sequence in (5.7) is a CC-addition chain for $2^n - 2$ (w.r.t. n), and hence $m_n(2^n - 2) = t$. □

Lemma 5.3. $m_9(508) = 4$.

Proof. From Proposition 5.1, we have $m_9(508) \geq \lceil \log_2(7) \rceil = 3$. We now rule out the possibility that $m_9(508) = 3$. Let there be a CC-addition chain for 508 (w.r.t. 9) of length 3. The only possibility is that in such a chain, the Hamming weight doubles after each of the first two (addition) steps. But in the last step, we must have two integers $a = \langle a_8 \dots a_0 \rangle_2$ and $b = \langle b_8 \dots b_0 \rangle_2$ such that $508 = a + b$, $\nu(a) = \nu(b)$, and both must come from the same cyclotomic class. Hence the bit-patterns of a and b must be cyclic shifts of each other. We just need to make sure that the bit-pattern $508 = \langle 111111100 \rangle_2$ cannot be obtained. There are four possible cases:

1. $a_0 = b_0 = 1$: then $a_1 = 1$ or $b_1 = 1$ (but not both). Hence with remaining 5 ones, it is not possible to obtain ones at the remaining 7 positions in the sum.
2. $a_0 = b_0 = 0$ and $a_1 = b_1 = 0$: now there are 8 ones for 7 positions. Hence a zero will appear in the sum when there is a one in the same position.
3. $a_0 = b_0 = 0$, $a_1 = b_1 = 1$ and $a_2 = b_2 = 1$: in this case it is not possible to get ones in 6 positions in the sum with only 4 ones.
4. $a_0 = b_0 = 0$, $a_1 = b_1 = 1$ and $a_2 = b_2 = 0$: by symmetry, we can set $a_3 = 1$ and $b_3 = 0$. Now there are 2 ones for a that can occur in any of the five remaining positions. Hence there are $\binom{5}{2} = 10$ choices. Once the two positions are fixed for a, then for b, the remaining three ones must be in the other three remaining positions of the sum. One can easily check in all the 10 cases that a and b are not cyclic shifts of each other.

Hence we obtain $m_9(508) > 3$. The CC-addition chain

$$\begin{aligned}\langle 1 \rangle_2 &\overset{\times}{\to} \langle 10 \rangle_2 \overset{+}{\to} \langle 11 \rangle_2 \overset{\times}{\to} \langle 1100 \rangle_2 \overset{+}{\to} \langle 1111 \rangle_2 \overset{\times}{\to} \langle 111100 \rangle_2 \\ &\overset{+}{\to} \langle 111111 \rangle_2 \overset{\times}{\to} \langle 1111110 \rangle_2 \overset{+}{\to} \langle 1111111 \rangle_2 \overset{\times}{\to} \langle 111111100 \rangle_2.\end{aligned}$$

shows that $m_9(508) \leq 4$. Hence $m_9(508) = 4$ □

5.2.4 CC-Addition Chain: Monotonicity

It is natural to ask how the value of $m_n(\alpha)$ varies with n. As mentioned previously, $m_n(\alpha)$ is defined only for $n \geq \lceil \log_2(\alpha + 2) \rceil$. Is the value of $m_n(\alpha)$ independent of n for a given value of α? This is not true since we have

already seen the counterexample $m_5(23) = 2$ but $m_6(23) = 3$. The example $m_7(83) = 3$ but $m_9(83) = 2$ shows that $m_n(\alpha)$ can also decrease as n increases. We can generalize the above examples to obtain infinitely many examples. For instance, consider $m_n\Big(\langle 1\underbrace{0\ldots0}_{n-4}111\rangle_2\Big) = m_n\Big(\langle \underbrace{0\ldots0}_{n-4}1111\rangle_2\Big) = 2$ but $m_{n+1}\Big(\langle 01\underbrace{0\ldots0}_{n-4}111\rangle_2\Big) = m_{n+1}\Big(\langle \underbrace{0\ldots0}_{n-4}11101\rangle_2\Big) = 3$, where $n \geq 5$.

But we can still show that $m_n(\alpha) \leq m_{n'}(\alpha)$ if $n \,|\, n'$, i.e. if n divides n'.

Theorem 5.1. *Let $\alpha, n, n' \in \mathbb{N}$, $n \,|\, n'$ and $\lceil \log_2(\alpha+2) \rceil \leq n \leq n'$. Then $m_n(\alpha) \leq m_{n'}(\alpha)$.*

Proof. The basic idea is to transform *any* CC-addition chain for α w.r.t. n' into a CC-addition chain for α w.r.t. n such that the length of the resulting chain is at most the length of the original one. This implies that $m_n(\alpha) \leq m_{n'}(\alpha)$. Let

$$C'_{b_0} = C'_1,\ C'_{b_1},\ C'_{b_2,}\ \cdots,\ C'_{b_r} = C'_\alpha \tag{5.9}$$

be a CC-addition chain for α w.r.t. n'. Let $a_i := b_i \pmod{2^n - 1}$ and C_{a_i} be the cyclotomic class of a_i w.r.t. n, $\forall i = 0, 1, \ldots, r$. Consider the sequence

$$C_{a_0},\ C_{a_1},\ C_{a_2,}\ \cdots,\ C_{a_r}. \tag{5.10}$$

The claim is that the above sequence in (5.10) is a CC-addition chain for α w.r.t. n. In particular, we need to prove that the sequence in (5.10) satisfies two properties. First is the CC-addition chain property (w.r.t. n), i.e. Definition 5.3, and the second one is $C_{a_r} = C_\alpha$ (w.r.t. n).

Claim 5.1 The sequence in (5.10) satisfies Definition 5.3 w.r.t. n.

Proof. First we need to show that the mapping $C'_{b_j} \mapsto C_{a_j}$ is well-defined. This is because the cyclotomic class C'_{b_j} may be represented as $C'_{\beta_{j''}}$, where $\beta_{j''} \in C'_{b_j}$. From Definition 5.3, $\beta_{j''} \in C'_{b_j}$ iff $\beta_{j''} = b_j \cdot 2^{j''} \pmod{2^{n'} - 1}$ for some $j'' \in \mathbb{N}$. Since $b_j \equiv a_j \pmod{2^n - 1}$, we have $\beta_{j''} \equiv b_j \cdot 2^{j''} \equiv a_j \cdot 2^k \pmod{2^n - 1}$, where $k := j'' \pmod{n}$. This proves the well-definedness property of the mapping of cyclotomic classes. Next, to prove the additivity property, observe that for every $i = 1, 2, \ldots, r$, there exist $0 \leq j, k < i$, $\beta_{i'} \in C'_{b_i}$, $\beta_{j'} \in C'_{b_j}$, and $\beta_{k'} \in C'_{b_k}$ such that $\beta_{i'} \equiv \beta_{j'} + \beta_{k'} \pmod{2^{n'} - 1}$. This is because the sequence in (5.9) is a CC-addition chain (w.r.t. n'). From the reasoning above, we can write $\beta_{i'} \equiv a_i \cdot 2^{i'} \pmod{2^n - 1}$, $\beta_{j'} \equiv a_j \cdot 2^{j'} \pmod{2^n - 1}$ and $\beta_{k'} \equiv a_k \cdot 2^{k'} \pmod{2^n - 1}$. Since $n \,|\, n'$, we have $2^n - 1 \,|\, 2^{n'} - 1$. Hence $\beta_{i'} \equiv \beta_{j'} + \beta_{k'} \pmod{2^n - 1}$ .Therefore, $a_i \cdot 2^{i'} \equiv a_j \cdot 2^{j'} + a_k \cdot 2^{k'} \pmod{2^n - 1}$. This proves the additivity property of the sequence in (5.10). □

Claim 5.2 $C_{a_r} = C_\alpha$.

Proof. Since $C'_{b_r} = C'_\alpha$ (w.r.t. n') from (5.9), we have $\alpha \equiv b_r 2^t \left(\mathrm{mod}\, 2^{n'} - 1\right)$ for some $t \in \mathbb{N}$ and $t < n'$. Since $2^n - 1 \,|\, 2^{n'} - 1$, we have $\alpha \equiv b_r 2^t \equiv a_r 2^{t'} \pmod{2^n - 1}$. Therefore, $C_{a_r} = C_\alpha$. □

This completes the proof of Theorem 5.1. □

Theorem 5.1 suggests that, to find a minimum length CC-addition chain w.r.t. n', first try to find one w.r.t. a divisor n of n'. Since $\mathbb{F}_{2^n}$ is a smaller field than $\mathbb{F}_{2^{n'}}$, it may be advantageous to work in $\mathbb{F}_{2^n}$. Once a minimum length CC-addition chain w.r.t. n is found, then check if it is a CC-addition chain w.r.t. n'. If it is the case, then it will be a minimum length chain.

5.3 Evaluation of Polynomials

5.3.1 $\mathbb{F}_{2^n}$-Polynomial Chain

The masking complexity of an S-box (Definition 5.5) corresponds to the min-length CC-addition chain of the exponent when it can be represented as a power. However when the S-box has a general polynomial representation, a notion similar to CC-addition chain is required. For evaluating polynomials (over $\mathbb{R}$) the notion of *polynomial chain* is given in [Knu97, Section 4.6.4]. In case of polynomials in $\mathbb{F}_{2^n}[x]$, we define the notion of $\mathbb{F}_{2^n}$*-polynomial chain*, where we do not count addition, scalar multiplication and squaring operations. Note that if $x, y \in \mathbb{F}_{2^n}$, then $x^{2^n} = x$ and $(x+y)^2 = x^2 + y^2$.

Definition 5.4. *A* $\mathbb{F}_{2^n}$-polynomial chain S *for a polynomial* $P(x) \in \mathbb{F}_{2^n}[x]$ *is defined as*

$$\lambda_{-1} = 1,\ \lambda_1 = x,\ \ldots\ ,\ \lambda_r = P(x) \tag{5.11}$$

where

$$\lambda_i = \begin{cases} \lambda_j + \lambda_k & -1 \le j, k < i, \\ \lambda_j \cdot \lambda_k & -1 \le j, k < i, \\ \alpha_i \odot \lambda_j & -1 \le j < i,\ \alpha_i \text{ is a scalar}, \\ \lambda_j^2 & -1 \le j < i. \end{cases}$$

Note that here $\cdot$ *and* $\odot$ *both perform the same operation, multiplication in* $\mathbb{F}_{2^n}$*. However in order to differentiate the non-linear operation we use* $\odot$ *for scalar multiplication. Here* $\lambda_j \cdot \lambda_k$ *denotes a non-linear multiplication. Let the number of non-linear multiplications involved in chain* S *be* $\mathcal{N}(S)$*. Then the* **non-linear complexity** *of* $P(x)$ *(over* $\mathbb{F}_{2^n}$*), denoted by* $\mathcal{M}(P(x))$*, is defined as* $\mathcal{M}(P(x)) = \min_S \mathcal{N}(S)$*, where* S *computes* $P(x)$*.*

5.3.2 Masking Complexity: Well-definedness

The *masking complexity* of an S-box is formally defined as follows.

Definition 5.5. [Masking Complexity] *Let $m, n \in \mathbb{N}$ with $m \leq n$. The* masking complexity *of an (n, m)-S-box is the non-linear complexity of $P(x)$, where $P(x)$ is the polynomial representation of the S-box over $\mathbb{F}_{2^n}$.*

Note that the above definition has been intuitively described in [CGP+12, Definition 1] as the minimum number of non-linear multiplications needed to evaluate the polynomial representation. Once the bit strings are identified naturally with the elements of $\mathbb{F}_{2^n}$ (given a field representation), then we can apply Lagrange interpolation technique to compute the (unique) polynomial of degree at most $2^n - 1$ representing the S-box in the corresponding field.

The well-definedness and relevance of the above definition of masking complexity is guaranteed because of the following reasons.

1. A natural question is - *does masking complexity change with the irreducible polynomial used to represent $\mathbb{F}_{2^n}$?* Note that under the natural mapping of bit strings to the field elements, the same S-box may correspond to different polynomials over $\mathbb{F}_{2^n}$ for different representations of the field. However we show in Theorem 5.2 that masking complexity does not depend on the field representation.
2. It is relatively straightforward to mask affine functions. In $\mathbb{F}_{2^n}$, squaring is linear, and affine functions are free from any "non-linear" multiplications.

The n-bit strings can be naturally mapped to field elements of $\mathbb{F}_{2^n}$ represented as polynomials over $\mathbb{F}_2$ modulo a degree n irreducible polynomial $f_1(y)$. Formally, $\mathcal{B}_1 : \{0,1\}^n \to \mathbb{F}_2[y]/f_1(y)$ is defined as

$$\mathcal{B}_1\left(\langle b_{n-1} b_{n-2} \dots b_0 \rangle\right) := \sum_{i=0}^{n-1} b_i \, y^i + \left(\mathbb{F}_2[y] \cdot f_1(y)\right), \tag{5.12}$$

where $b_i \in \{0,1\}$. The m-bit strings ($m \leq n$) are appended with leading zeros to identify them with n-bit strings. Later we shall see that it suffices if $\mathcal{B}_1$ is some $\mathbb{F}_2$-linear bijection. Note that $(\{0,1\}^n, \oplus)$ may be viewed as a vector space over $\mathbb{F}_2$.

Remark 5.2 It was claimed in [CGP+12, Remark 3] that the property of independence of masking complexity w.r.t. the irreducible polynomial used to represent $\mathbb{F}_{2^n}$ follows from the fact that field isomorphisms are $\mathbb{F}_2$-linear bijections. This reason is not enough and a formal proof requires more arguments, as we shall see in the proof of Theorem 5.2.

Let $f_1(y)$ and $f_2(z)$ be two irreducible polynomials of degree n over $\mathbb{F}_2$. Then $\mathbb{F}_2[y]/f_1(y)$ and $\mathbb{F}_2[z]/f_2(z)$ are two representations for $\mathbb{F}_{2^n}$. Let $\mathcal{B}_1 : \{0,1\}^n \to$

$\mathbb{F}_2[y]/f_1(y)$ be as in (5.12), and $\mathcal{B}_2 : \{0,1\}^n \to \mathbb{F}_2[z]/f_2(z)$ be analogously defined for $f_2(z)$. Note that $\mathcal{B}_1$ and $\mathcal{B}_2$ are $\mathbb{F}_2$-linear isomorphisms between vector spaces. The corresponding inverse maps $\mathcal{B}_1^{-1}$ and $\mathcal{B}_2^{-1}$ are also $\mathbb{F}_2$-linear isomorphisms of vector spaces.

Let $\mathcal{U} : \{0,1\}^n \to \{0,1\}^n$ be any function on n-bit strings. For instance, $\mathcal{U}$ may represent an (n,m)-S-box (upon padding m-bit strings with leading zeros). The maps $\mathcal{U}$ and $\mathcal{B}_1$ will "induce" a map $\mathcal{U}_1 : \mathbb{F}_2[y]/f_1(y) \to \mathbb{F}_2[y]/f_1(y)$. More precisely,

$$\mathcal{U}_1 = \mathcal{B}_1 \circ \mathcal{U} \circ \mathcal{B}_1^{-1}. \tag{5.13}$$

Similarly we can define

$$\mathcal{U}_2 = \mathcal{B}_2 \circ \mathcal{U} \circ \mathcal{B}_2^{-1}. \tag{5.14}$$

Let $P_1(x)$ and $P_2(x)$ be the polynomial representations (of degree at most $2^n - 1$) of $\mathcal{U}_1$ and $\mathcal{U}_2$, respectively. We now prove the following theorem.

Theorem 5.2. *$\mathcal{M}(P_1(x)) = \mathcal{M}(P_2(x))$, where $P_1(x)$ and $P_2(x)$ are as defined above. In other words, the masking complexity of an S-box (in general, any function on bit strings) is invariant w.r.t. field representations.*

Proof. Let the maps $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{U}$, $\mathcal{U}_1$ and $\mathcal{U}_2$ be as defined in (5.13) and (5.14). Since two finite fields of the same order are isomorphic, there exists a field isomorphism $\psi : \mathbb{F}_2[y]/f_1(y) \to \mathbb{F}_2[z]/f_2(z)$. Note that the map ψ is also an $\mathbb{F}_2$-linear isomorphism between vector spaces that is compatible with the multiplication operation of the fields. Let $\mathcal{H} : \mathbb{F}_2[y]/f_1(y) \to \mathbb{F}_2[z]/f_2(z)$ be defined as

$$\mathcal{H} = \mathcal{B}_2 \circ \mathcal{B}_1^{-1}. \tag{5.15}$$

Since $\mathcal{B}_1$ and $\mathcal{B}_2$ are $\mathbb{F}_2$-linear bijections, so will be $\mathcal{H}$. Note that $\mathcal{H}$ need not be a field isomorphism. Also define the maps $\mathcal{H}^*, \mathcal{U}_1^* : \mathbb{F}_2[z]/f_2(z) \to \mathbb{F}_2[z]/f_2(z)$ as

$$\mathcal{H}^* = \mathcal{H} \circ \psi^{-1}, \tag{5.16}$$

$$\mathcal{U}_1^* = \psi \circ \mathcal{U}_1 \circ \psi^{-1}. \tag{5.17}$$

Intuitively, the maps $\mathcal{H}^*$ and $\mathcal{U}_1^*$ are analogues of $\mathcal{H}$ and $\mathcal{U}_1$ that are maps from $\mathbb{F}_2[z]/f_2(z)$ to itself. From (5.13), (5.14), (5.15) and (5.17), we have

$$\mathcal{U}_1 = \psi^{-1} \circ \mathcal{U}_1^* \circ \psi = \mathcal{H}^{-1} \circ \mathcal{U}_2 \circ \mathcal{H}.$$

Hence from (5.17), we get

$$\mathcal{U}_2 = \mathcal{H}^* \circ \mathcal{U}_1^* \circ \mathcal{H}^{*-1}. \tag{5.18}$$

Let $P_{\mathcal{H}^*}(x)$, $P_{\mathcal{H}^{*-1}}(x)$ and $P_{\mathcal{U}_1^*}(x)$ be polynomials over $\mathbb{F}_2[z]/f_2(z)$ of degree at most $2^n - 1$ representing $\mathcal{H}^*$, $\mathcal{H}^{*-1}$ and $\mathcal{U}_1^*$, respectively. From the above relation, we obtain

$$P_2(x) = P_{\mathcal{H}^*}\left(P_{\mathcal{U}_1^*}\left(P_{\mathcal{H}^{*-1}}(x)\right)\right). \tag{5.19}$$

It is precisely to get the above relation that we had to introduce the maps $\mathcal{H}^*$ and $\mathcal{U}_1^*$. The following two lemmas show that $\mathcal{M}\left(P_{\mathcal{U}_1^*}(x)\right) = \mathcal{M}\left(P_1(x)\right)$ and $\mathcal{M}\left(P_{\mathcal{H}^{*-1}}\right) = \mathcal{M}\left(P_{\mathcal{H}^*}\right) = 0$.

Lemma 5.4. $\mathcal{M}\left(P_{\mathcal{U}_1^*}(x)\right) = \mathcal{M}\left(P_1(x)\right)$.

Proof. Let $P_1(x) = \sum_{i=0}^{2^n-1} a_i\, x^i$, where $a_i \in \mathbb{F}_2[z]/f_2(z)$. From the definition of $\mathcal{U}_1^*$ in (5.17), it follows that $P_{\mathcal{U}_1^*}(x) = \sum_{i=0}^{2^n-1} \psi(a_i)\, x^i$. Using the field isomorphisms ψ and ψ^{-1}, any polynomial chain to evaluate $P_1(x)$ can be converted to one that evaluates $P_{\mathcal{U}_1^*}(x)$, and vice-versa. Hence the lemma follows. □

Lemma 5.5. *Let $\mathcal{A} : \mathbb{F}_2[z]/f_2(z) \to \mathbb{F}_2[z]/f_2(z)$ be an $\mathbb{F}_2$-affine function and $P_{\mathcal{A}}(x)$ be the corresponding polynomial representation of degree at most $2^n - 1$. Then $P_{\mathcal{A}}(x) = \sum_{i=0}^{n-1} a_i\, x^{2^i} + a_{-1}$, for some $a_i \in \mathbb{F}_2[z]/f_2(z)$ $(-1 \le i \le n-1)$, and $\mathcal{M}\left(P_{\mathcal{A}}(x)\right) = 0$.*

Proof. Since $\mathcal{A}$ is an $\mathbb{F}_2$-affine function, it can be written as $\mathcal{A} = \mathcal{A}' + a_{-1}$, where $\mathcal{A} : \mathbb{F}_2[z]/f_2(z) \to \mathbb{F}_2[z]/f_2(z)$ is an $\mathbb{F}_2$-linear map, and $a_{-1} \in \mathbb{F}_2[z]/f_2(z)$. It is enough to show that the polynomial $P_{\mathcal{A}'}(x)$ corresponding to $\mathcal{A}'$ is of the form $\sum_{i=0}^{n-1} a_i\, x^{2^i}$. Suppose that there exists a term x^m in $P_{\mathcal{A}'}(x)$ whose coefficient is non-zero, and $m \neq 2^j$ for $0 \le j \le n-1$. Let x^m be the largest among such terms and write $m = 2^t \cdot k$, where $k > 1$ is odd. Define the bivariate polynomial

$$P'(x, y) = P_{\mathcal{A}'}(x + y) - P_{\mathcal{A}'}(x) - P_{\mathcal{A}'}(y).$$

We have $P'(x, y) \not\equiv 0$ since the coefficient of the term $x^{2^t(k-1)} \cdot y^{2^t}$ is 1. This can be easily seen from the fact that $(x+y)^{2^t k} = \left(x^{2^t} + y^{2^t}\right)^k$, and then using the binomial expansion and the fact that the characteristic of $\mathbb{F}_{2^n}$ is two. By the linearity of $\mathcal{A}'$, we require $P'(\alpha, \beta) = 0$ for all $\alpha, \beta \in \mathbb{F}_2[z]/f_2(z)$. But this is not possible since the total degree of $P'(x, y)$ is $m < 2^n$, and from the Schwartz-Zippel lemma (cf. Appendix 2.2), the polynomial $P'(x, y)$ can have at most $m \cdot 2^n$ roots. Hence $P'(x, y) \equiv 0$ and $P_{\mathcal{A}'}(x) = \sum_{i=0}^{n-1} a_i\, x^{2^i}$ and $\mathcal{M}\left(P_{\mathcal{A}'}(x)\right) = \mathcal{M}\left(P_{\mathcal{A}}(x)\right) = 0$. By the linearity of $P_{\mathcal{A}'}(x)$, we have $P_{\mathcal{A}'}(0) = 0$. Hence the lemma follows. □

From (5.19), Lemma 5.4 and Lemma 5.5, we have $\mathcal{M}\left(P_2(x)\right) \le \mathcal{M}\left(P_1(x)\right)$. From (5.18), we get $\mathcal{U}_1^* = \mathcal{H}^{*-1} \circ \mathcal{U}_2 \circ \mathcal{H}^*$. Hence $\mathcal{M}\left(P_1(x)\right) \le \mathcal{M}\left(P_2(x)\right)$. Therefore, $\mathcal{M}\left(P_1(x)\right) = \mathcal{M}\left(P_2(x)\right)$. This completes the proof of Theorem 5.2. □

Lemma 5.6. [CGP+12, Proposition 1] *The masking complexity of an S-box (in general, any function) cannot increase when it is composed with affine functions. When composed with affine bijections, then masking complexity remains the same.*

Remark 5.3 Note that Lemma 5.6 holds only when the evaluation of affine functions over $\mathbb{F}_{2^n}$ does not involve any non-linear multiplication. For the sake of completeness, this property is proved in Lemma 5.5.

Note that in the proof of Theorem 5.2 the only property of the maps $\mathcal{B}_1$ and $\mathcal{B}_2$ used is that they are $\mathbb{F}_2$-linear bijections. Hence if $\mathcal{B}_1$ and $\mathcal{B}_2$ are any linear bijections, even then the masking complexity of an S-box remains invariant.

5.3.3 Non-linear Complexity: Lower Bounds

We now give lower bounds on the non-linear complexity of evaluating polynomials over $\mathbb{F}_{2^n}$. Our first lower bound result originally appeared in [RV13]. Later we improved this result significantly in [CRV14].

5.3.3.1 First Bound

Proposition 5.3. *Let $P(x) := \sum_{i=0}^{2^n-1} a_i\, x^i$ be a polynomial in $\mathbb{F}_{2^n}[x]$. Then*

$$\mathcal{M}(P(x)) \geq \max_{\substack{0<i<2^n-1 \\ a_i \neq 0}} m_n(i).$$

Proof. To prove the proposition, we just need to prove the following claim. Let $\sigma_k^n := \{\alpha \,|\, m_n(\alpha) \leq k\}$. We claim that, with at most k non-linear multiplications, we can evaluate only those polynomials of the form $\sum_i a_i x^i$, where $i \in \sigma_k^n$ and $a_i \in \mathbb{F}_{2^n}$. It is easy to see that with zero non-linear multiplications, only those polynomials of the form $\sum_i a_i x^i$, where $i \in \sigma_0^n = \{2^j | 0 \leq j \leq n-1\}$. Let us assume that the above claim is true up to $k-1$ non-linear multiplications. Consider the set of polynomials $T := \left\{p(x) \mid p(x) = \sum_j b_j x^j,\ j \in \sigma_{k-1}^n,\ b_j \in \mathbb{F}_{2^n}\right\}$. Since squaring is a linear operation in $\mathbb{F}_{2^n}[x]$, the set T is closed under additions, scalar multiplications and squaring operations. Hence if we allow only one more non-linear multiplication, then exponents in the resulting polynomial can only be from σ_k^n. Note that $m_n(\alpha)$ is defined only for $0 < \alpha < 2^n - 1$ and $x^{2^n-1} = 1$ if $x \neq 0$. This proves the claim. □

5.3.3.2 Improved Bound

Our technique to prove the lower bound of $\Omega(\sqrt{2^n/n})$ on the non-linear complexity is similar to the one used in the proof of [PS73, Theorem 2]. But we would like to emphasize that their result is not applicable to our setting since they work over the integers and the cost model used there is different from the one used in our case.

Proposition 5.4. *There exists a polynomial $P(x) \in \mathbb{F}_{2^n}[x]$ such that $\mathcal{M}(P(x)) \geq \sqrt{\frac{2^n}{n}} - 2$.*

Proof. At a more abstract level, an $\mathbb{F}_{2^n}$-polynomial chain evaluating $P(x) \in \mathbb{F}_{2^n}[x]$ that uses r non-linear multiplications ($r \geq 0$) can be equivalently described as a sequence $\mathcal{Z}$ of polynomials z_{-1}, z_0, ..., $z_r, P(x)$, where

$$\begin{aligned} z_{-1} &= 1, \\ z_0 &= x, \\ z_k &= \left(\beta_{k,-1} + \sum_{i=0}^{k-1}\sum_{j=0}^{n-1} \beta_{k,i,j}\, z_i^{2^j}\right) \cdot \left(\beta'_{k,-1} + \sum_{i=0}^{k-1}\sum_{j=0}^{n-1} \beta'_{k,i,j}\, z_i^{2^j}\right) \\ &\qquad (\text{mod } x^{2^n} + x), \quad (5.20) \end{aligned}$$

where $k = 1, 2, \ldots, r$, $\beta_{k,-1}, \beta'_{k,-1}, \beta_{k,i,j}, \beta'_{k,i,j} \in \mathbb{F}_{2^n}$. Lastly,

$$P(x) = \beta_{r+1,-1} + \sum_{i=0}^{r}\sum_{j=0}^{n-1} \beta_{r+1,i,j}\, z_i^{2^j} \pmod{x^{2^n} + x}, \qquad (5.21)$$

where again $\beta_{r+1,-1}, \beta_{r+1,i,j} \in \mathbb{F}_{2^n}$.

Since the squaring operation is $\mathbb{F}_2$-linear in $\mathbb{F}_{2^n}$, and that $x^{2^n} = x$ for all $x \in \mathbb{F}_{2^n}$, it is easy to see that any polynomial that can be evaluated using at most r non-linear multiplications will be of the form as given in (5.21).

The number of parameters $\beta_{k,-1}$, $\beta'_{k,-1}$, $\beta_{k,i,j}$, $\beta'_{k,i,j}$ in (5.20) for a given value of k ($k = 1, \ldots, r$) is $2 \cdot (k \cdot n + 1)$. In (5.21), the number of parameters $\beta_{r+1,-1}$, $\beta_{r+1,i,j}$ is $(r+1) \cdot n + 1$. Totally, the number of parameters are

$$(r+1)\, n + 1 + \sum_{k=1}^{r} 2\,(kn+1)\,.$$

Since there are only $|\mathbb{F}_{2^n}|^{2^n}$ distinct polynomials in $\mathbb{F}_{2^n}[x]$ (i.e. up to evaluation), and a given set of values for the parameters enables to evaluate a single polynomial only, we get the following necessary condition to evaluate all polynomials over $\mathbb{F}_{2^n}[x]$

$$\begin{aligned} |\mathbb{F}_{2^n}|^{(r+1)n+1+\sum_{k=1}^{r} 2(kn+1)} &\geq |\mathbb{F}_{2^n}|^{2^n}, \\ \Longrightarrow \quad (r+1)\, n + 1 + \sum_{k=1}^{r} 2\,(kn+1) &\geq 2^n, \\ \Longrightarrow \quad n \cdot r^2 + (2n+2) \cdot r - (2^n - n - 1) &\geq 0, \\ \Longrightarrow \quad r &\geq \sqrt{\frac{2^n}{n}} - 2. \qquad (5.22) \end{aligned}$$

Hence there exists polynomials over $\mathbb{F}_{2^n}$ that require $\Omega(\sqrt{2^n/n})$ non-linear multiplications to evaluate them. □

Concrete Lower Bounds. In Table 5.2 we compare, for various values of n, the previously known lower bound for non-linear complexity with the new lower bound as determined by (5.22).

n	4	5	6	7	8	9	10	11	12
Previous/our lower bound (cf. [CGP$^+$12], Prop. 5.3)	2	2	3	3	4	4	4	4	4
Our lower bound (cf. (5.22))	**0**	**1**	**2**	**3**	**4**	**6**	**9**	**12**	**17**

Table 5.2: Lower bound for non-linear complexity in $\mathbb{F}_{2^n}$.

5.3.3.3 Lower Bounds for S-boxes

Though we defer the discussion on the application to S-boxes to Section 5.4, for clarity, we briefly mention the application of the above lower bound results here.

We represent the fields $\mathbb{F}_{2^4}$, $\mathbb{F}_{2^6}$ and $\mathbb{F}_{2^8}$ using irreducible polynomials $y^4 + y^3 + 1$, $y^6 + y^4 + y^3 + y + 1$, $y^8 + y^4 + y^3 + y + 1 \in \mathbb{F}_2[y]$, respectively. From Theorem 5.2, we know that the masking complexity is invariant w.r.t. the field representations.

The polynomials corresponding to the eight DES S-boxes are polynomials of degree 62 in $\mathbb{F}_{2^6}[x]$ (here the 4 bit DES S-box outputs are padded with two leading zeroes and identified with the elements of $\mathbb{F}_{2^6}$). For the PRESENT S-box, the corresponding polynomial is a polynomial of degree 14 in $\mathbb{F}_{2^4}[x]$. Since $m_6(62) = 3$ and $m_4(14) = 2$, from Proposition 5.3 we obtain the following corollary.

Corollary 5.1. *Masking complexity of a DES S-box is at least* 3*, and that of the PRESENT S-box is at least* 2*.*

The AES S-box can be written as an affine permutation composed with the polynomial $x^{254} \in \mathbb{F}_{2^8}[x]$. From Lemma 5.6, the masking complexity of AES S-box is $\mathcal{M}\left(x^{254}\right)$ over $\mathbb{F}_{2^8}$. Using arguments similar to the proof of Lemma 5.3, we obtain the following corollary.

Corollary 5.2. *Masking complexity of the AES S-box is at least 4.*

The above corollary was shown by exhaustive search in [CGP$^+$12].

5.3.4 Generic Polynomial Evaluation Technique

Next we shall look at various methods to efficiently evaluate general polynomials over $\mathbb{F}_{2^n}$. As expected, the cost model we use is the non-linear complexity. First, we briefly recollect the previously known methods for this problem. Then we present a new improved method that is (heuristically) optimal. This method originally appeared in [CRV14].

Remark 5.4 Since our goal is only to evaluate polynomials over $\mathbb{F}_{2^n}$, we will be actually working in the ring $\mathbb{F}_{2^n}[x]/(x^{2^n}+x)$, which is an abuse of the notation $\mathbb{F}_{2^n}[x]$. In other words, we treat any given polynomial $P(x) = \sum_{i=0}^{2^n-1} a_i x^i \in \mathbb{F}_{2^n}[x]$ to be the same as $P(x)$ modulo $x^{2^n}+x$; hence $P(x)$ has degree at most $N := 2^n - 1$.

5.3.4.1 Previous Generic Methods

Horner's Method. This is a classical and one of the simplest methods used to evaluate a given polynomial. The polynomial $P(x)$ is computed as

$$P(x) = ((((a_N\, x + a_{N-1})\, x + a_{N-2})\, x) + \ldots + a_1)\, x + a_0.$$

This method does not take advantage of the fact that squarings are free. The number of non-linear multiplications required is one less than the number of monomials that are non-constant and have non-zero coefficients. The number of scalar multiplications required is 1. The number of additions required is one less than the number of monomials having non-zero coefficients.

In the worst case (for dense polynomials), the number of non-linear multiplications required is $2^n - 2$. The number of additions required is $2^n - 1$.

Cyclotomic-Class Method. This method, proposed in [CGP+12], is an optimized version of the brute force evaluation method of computing every monomial, then multiplying with the corresponding coefficients and then summing the terms. The basic idea is to use one non-linear multiplication to compute the monomial x^{α_j} for a new cyclotomic class $\alpha_j \in C_{\alpha_j}$. Now for every other element in C_{α_j}, we can compute the corresponding monomial for free by squaring. Once all the monomials are computed, then using only scalar multiplications and additions, we can compute $P(x)$.

In the worst case, the number of non-linear multiplications required is same as the number of cyclotomic classes, which is at least $\frac{2^n-1}{n}$ (cf. Remark 5.1). The number of squarings required is about $2^n - 1$ less the number of cyclotomic classes, while the actual number of scalar multiplications required is the number of non- monic coefficients less one (for the constant term) (hence at most $2^n - 1$), and the number of additions required is one less than the number of monomials having non-zero coefficients (hence at most $2^n - 1$).

Parity-Split Method. This method was also proposed in [CGP+12]. It is based on the idea of representing $P(x)$ in terms of polynomials having only monomials raised to even powers:

$$P(x) = P_{1,1}(x^2) + x \cdot P_{1,2}(x^2),$$

where $P_{1,1}$ and $P_{1,2}$ are polynomials of degree at most $N/2$. On recursively applying this technique, we can express $P(x)$ in terms of polynomials successively having half the degree of those in the previous step. Note that at every step the number of non-linear multiplications required at the combining step also doubles compared with the previous level. Optimal depth is $\lfloor n/2 \rfloor$ (i.e., after $\lfloor n/2 \rfloor$ recursive decompositions of $P(x)$), and the required number of non-linear multiplications in the worst case is approximately $\sqrt{2} \cdot 2^{n/2}$.

The number of scalar multiplications and additions required is the same as that of the cyclotomic-class method. The number of squarings for the parity-split method is about $2^{\lceil n/2 \rceil - 1} + \lfloor n/2 \rfloor$ (for dense polynomials).

Divide-and-Conquer Method. This method was originally proposed in [PS73]. It is based on the idea of expressing the given polynomial in terms of polynomials of smaller degree. This method is effective if the degree of the given polynomial if of specific form, as discussed below.

Let $P(x)$ be a polynomial having degree $N = k(2t-1)$. We divide $P(x)$ by x^{kt} and express $P(x)$ as following

$$P(x) = Q(x) \cdot x^{kt} + R(x) \tag{5.23}$$

where Q is monic and $\deg(Q) = k(t-1)$, $\deg(R) \leq kt - 1$. Now we divide $R(x) - x^{k(t-1)}$ by $Q(x)$ and obtain $C(x)$, $R_1(x)$ as following

$$R(x) - x^{k(t-1)} = C(x) \cdot Q(x) + R_1(x) \tag{5.24}$$

where $\deg(C) \leq k-1$, $\deg(R_1) \leq k(t-1) - 1$. So $P(x)$ can be written as

$$P(x) = (x^{kt} + c(x)) \cdot Q(x) + x^{k(t-1)} + R_1(x) \tag{5.25}$$

Note that $(x^k)^t + c(x))$ is already a function of polynomials having degree at most k. Assume that $t = 2^{i-1}$, then having computed $x^2, x^3, ..., x^k$ we can compute x^{kt} for "free"(without non-linear multiplications).

Next we apply the same technique to $Q(x)$ and $x^{k(t-1)} + R_1(x)$ (both having degree $k(t-1)$) recursively. In general, if $i \leq m$ then the number of non-linear multiplications can be calculated from the relation

$$\mathcal{T}(k(2^i - 1)) = 2\mathcal{T}(k(2^{i-1} - 1)) + 1 \tag{5.26}$$

where $\mathcal{T}(\gamma)$ is the number of non-linear multiplications required to evaluate a polynomial having degree γ, using the above technique. This gives $\mathcal{T}(k(2^m - 1)) = 2^{m-1} - 1 \approx N/2k$. Hence the total number of non-linear multiplications is about $\frac{1}{2}(k + N/k)$.

If the degree $N \approx k \cdot (2^m - 1)$, then the number of additions is about $(k + 1) \cdot (2^m - 1)$. The number of non-linear multiplications is about $k \cdot (2^m - 1)$. The number of squarings is about $\frac{k}{2} + \log_k d$. Hence if $k \approx \sqrt{d}$, then this is

about $\frac{\sqrt{d}}{2} + 2$. Hence for dense polynomials, there is no significant overhead with respect to the linear operations.

As we shall see in Section 5.4.4.1, we can suitably adapt this technique (at least to some cases of practical relevance for S-boxes) even when the degree of the polynomial is not of specific form as required.

5.3.4.2 New Generic Method

Description. Consider an n-bit to n-bit S-Box represented by a polynomial $P(x) \in \mathbb{F}_{2^n}[x]$. We consider a collection $\mathcal{S}$ of ℓ cyclotomic classes w.r.t. n:

$$\mathcal{S} = \{C_{\alpha_1=0}, C_{\alpha_2=1}, C_{\alpha_3}, \ldots, C_{\alpha_\ell}\}\,. \tag{5.27}$$

Also, define L as the set of all integers in the cyclotomic classes of $\mathcal{S}$:

$$L = \underset{C_i \in \mathcal{S}}{\cup} C_i. \tag{5.28}$$

We choose the set $\mathcal{S}$ of ℓ cyclotomic classes in (5.27) so that the set of corresponding monomials x^L from $\mathcal{S}$ can be computed using only $\ell - 2$ non-linear multiplications. We require that every monomial x^0, x^1, ... , x^{2^n-1}, can be written as product of some two monomials in $\mathcal{P}(x^L)$. Moreover, we try to choose only those cyclotomic classes with the maximum number of n elements (except C_0 which has only a single element). This gives

$$|L| = 1 + n \cdot (\ell - 1)\,. \tag{5.29}$$

Next, we generate $t - 1$ random polynomials $q_i(x) \xleftarrow{\$} \mathcal{P}(x^L)$ that have their monomials only in x^L. Suitable values for the parameters t and $|L|$ will be determined later. Then, we try to find t polynomials $p_i(x) \in \mathcal{P}(x^L)$ such that

$$P(x) = \sum_{i=1}^{t-1} p_i(x) \cdot q_i(x) \; + \; p_t(x). \tag{5.30}$$

It is easy to see that the coefficients of the $p_i(x)$ polynomials can be obtained by solving a system of linear equations in $\mathbb{F}_{2^n}$, as in the Lagrange interpolation theorem. More precisely, to find the polynomials $p_i(x)$, we solve the following system of linear equations over $\mathbb{F}_{2^n}$:

$$A \cdot \vec{c} = \vec{b} \tag{5.31}$$

where the matrix A is obtained by evaluating the R.H.S. of (5.30) at every element of $\mathbb{F}_{2^n}$, and by treating the unknown coefficients of $p_i(x)$ as variables. This matrix has 2^n rows and $t \cdot |L|$ columns, since each of the t polynomials

$p_i(x)$ has $|L|$ unknown coefficients. The matrix A can also be written as a block concatenation of smaller matrices:

$$A = (A_1|A_2|\ldots|A_t), \tag{5.32}$$

where A_i is a $2^n \times |L|$ matrix corresponding to the product $p_i(x) \cdot q_i(x)$. Let $a_j \in \mathbb{F}_{2^n}$ $(j = 0, 1, \ldots, 2^n - 1)$ be all the field elements and $p_i(x)$ consists of the monomials $x^{k_1}, x^{k_2}, \ldots, x^{k_{|L|}} \in x^L$. Then, the matrix A_i has the following structure:

$$A_i = \begin{pmatrix} a_0^{k_1} \cdot q_i(a_0) & a_0^{k_2} \cdot q_i(a_0) & \ldots & a_0^{k_{|L|}} \cdot q_i(a_0) \\ a_1^{k_1} \cdot q_i(a_1) & a_1^{k_2} \cdot q_i(a_1) & \ldots & a_1^{k_{|L|}} \cdot q_i(a_1) \\ a_2^{k_1} \cdot q_i(a_2) & a_2^{k_2} \cdot q_i(a_2) & \ldots & a_2^{k_{|L|}} \cdot q_i(a_2) \\ \vdots & \vdots & \ldots & \vdots \\ a_{2^n-1}^{k_1} \cdot q_i(a_{2^n-1}) & a_{2^n-1}^{k_2} \cdot q_i(a_{2^n-1}) & \ldots & a_{2^n-1}^{k_{|L|}} \cdot q_i(a_{2^n-1}) \end{pmatrix} \tag{5.33}$$

The unknown vector $\vec{c}$ in (5.31) corresponds to the unknown coefficients of the polynomials $p_i(x)$. The vector $\vec{b}$ is formed by evaluating $P(x)$ at every element of $\mathbb{F}_{2^n}$. Note that since $P(x)$ corresponds to an S-box, the vector $\vec{b}$ can be directly obtained from the corresponding S-box lookup table.

If the matrix A has rank 2^n, then we are able to guarantee that the decomposition in (5.30) exists for every polynomial $P(x)$. To be of full rank 2^n the matrix must have a number of columns $\geq 2^n$. This gives us the necessary condition

$$t \cdot |L| \geq 2^n. \tag{5.34}$$

We stress that (5.34) is only a necessary condition. Namely we don't know how to prove that the matrix A will be full rank when the previous condition is satisfied; this makes our algorithm heuristic. In practice for random polynomials $q_i(x)$ we almost always obtain a full rank matrix under condition (5.34).

From (5.29), we get the condition

$$t \cdot (1 + n \cdot (\ell - 1)) \geq 2^n \tag{5.35}$$

where t is the number of polynomials $p_i(x)$ and ℓ the number of cyclotomic classes in the set $\mathcal{S}$, to evaluate a polynomial $P(x)$ over $\mathbb{F}_{2^n}$.

We summarize the above method in Algorithm 2 below. The number of non-linear multiplications required in the combining step (5.30) is $t - 1$. As mentioned earlier, we need $\ell - 2$ non-linear multiplications to precompute the set x^L. Hence the total number of non-linear multiplications required is then

$$N_{mult} = \ell - 2 + t - 1 = \ell + t - 3. \tag{5.36}$$

Algorithm 2 New generic polynomial decomposition algorithm

Input: $P(x) \in \mathbb{F}_{2^n}[x]$.

Output: Polynomials $p_i(x), q_i(x)$ such that $P(x) = \sum_{i=1}^{t-1} p_i(x) \cdot q_i(x) + p_t(x)$.

1: Choose ℓ cyclotomic classes C_{α_i} : $L \leftarrow \bigcup_{i=1}^{l} C_{\alpha_i}$, and the basis set x^L can be computed using $\ell - 2$ non-linear multiplications.
2: Choose t such that $t \cdot |L| \geq 2^n$.
3: For $1 \leq i \leq t$, choose $q_i(x) \xleftarrow{\$} \mathcal{P}\left(x^L\right)$.
4: Construct the matrix $A \leftarrow (A_1|A_2|\ldots|A_t)$, where each A_i is the $2^n \times |L|$ matrix given by (5.33).
5: Solve the linear system $A \cdot \vec{c} = \vec{b}$, where $\vec{b}$ is the evaluation of $P(x)$ at every element of $\mathbb{F}_{2^n}$.
6: Construct the polynomials $p_i(x)$ from the solution vector $\vec{c}$.

where t is the number of polynomials $p_i(x)$ and ℓ the number of cyclotomic classes in the set $\mathcal{S}$.

Remark 5.5 If A has rank 2^n, then the same set of basis polynomials $q_i(x)$ will yield a decomposition as in (5.30) for *any* polynomial $P(x)$. That is, the matrix A is independent from the polynomial $P(x)$ to be evaluated.

Remark 5.6 Our decomposition method is heuristic because for a given n in $\mathbb{F}_{2^n}$ we do not know how to guarantee that the matrix A has full rank 2^n. However for typical values of n, say $n = 4, 6, 8$, we can definitely check that the matrix A has full rank, for a particular choice of random polynomials $q_i(x)$. Then any polynomial $P(x)$ can be decomposed using these polynomials $q_i(x)$. In other words for a given n we can once and for all generate the random polynomials $q_i(x)$ and check that the matrix A has full rank 2^n, which will prove that any polynomial $P(x) \in \mathbb{F}_{2^n}[x]$ can then be decomposed as above. In summary our method is heuristic for large values of n, but can be proven for small values of n. Such proof requires to compute the rank of a matrix with 2^n rows and a slightly larger number of columns, which takes $\mathcal{O}(2^{3n})$ time using Gaussian elimination.

Asymptotic Analysis. Substituting (5.36) in (5.35) to eliminate the parameter ℓ, we get

$$t \cdot (1 + n \cdot (N_{mult} - t + 2)) \geq 2^n,$$

$$\implies \quad N_{mult} \geq \frac{2^n}{n \cdot t} + t - \left(2 + \frac{1}{n}\right). \tag{5.37}$$

The R.H.S. of the above expression is minimized when $t \approx \sqrt{\frac{2^n}{n}}$, and hence we obtain

$$N_{mult} \geq 2 \cdot \sqrt{\frac{2^n}{n}} - \left(2 + \frac{1}{n}\right). \tag{5.38}$$

Hence, our heuristic method requires $\mathcal{O}(\sqrt{2^n/n})$ non-linear multiplications, which is asymptotically slightly better than the Parity-Split method [CGP+12], which has proven complexity $\mathcal{O}(\sqrt{2^n})$. If one has to rigorously establish the above bound for our method, then we may have to prove the following statements, which we leave as open problems:

- We can sample the collection S of cyclotomic classes in (5.27), each having maximal length n (other than C_0), using at most $\ell - 2$ non-linear multiplications.
- The condition $t \cdot |L| \geq 2^n$ suffices to ensure that the matrix A has full rank 2^n.

Table 5.3 lists the expected minimum number of non-linear multiplications, as determined by (5.38), for binary fields $\mathbb{F}_{2^n}$ of practical interest. It also lists the actual number of non-linear multiplications that suffices to evaluate any polynomial, for which we have verified that the matrix A has full rank 2^n, for a particular random choice of the $q_i(x)$ polynomials. We also provide a performance comparison of our method with that of the Cyclotomic Class and the Parity-Split methods from [CGP+12]. In Table 5.4, we list the specific choice of parameters t and L that we used in this experiment.

n	4	5	6	7	8	9	10
Cyclotomic Class method [CGP+12]	3	5	11	17	33	53	105
Parity-Split method [CGP+12]	4	6	10	14	22	30	46
Expected minimum value of N_{mult} (cf. (5.38))	2	3	5	7	10	13	19
Achievable value of N_{mult}	**2**	**4**	**5**	**7**	**10**	**14**	**19**

Table 5.3: Minimum values of N_{mult}

Remark 5.7 Note that there is still a gap between the lower bound from Table 5.2 and the achievable value of N_{mult} for our method in Table 5.3. This is because in our method the decomposition of $P(x)$ as

$$P(x) = \sum_{i=1}^{t-1} p_i(x) \cdot q_i(x) + p_t(x) \tag{5.39}$$

is performed by first generating the polynomials $q_i(x)$ randomly and independently of $P(x)$, in order to have a linear system of equations over the coefficients of $p_i(x)$. Instead one could try to solve (5.39) for both the $p_i(x)$ and the $q_i(x)$ polynomials simultaneously; however this gives a quadratic system of equations, which seems much harder to solve.

n	t	L	$\lvert L\rvert$
4	2	$C_0 \cup C_1 \cup C_3$	9
5	3	$C_0 \cup C_1 \cup C_3 \cup C_7$	16
6	3	$C_0 \cup C_1 \cup C_3 \cup C_7 \cup C_{11}$	25
7	4	$C_0 \cup C_1 \cup C_3 \cup C_7 \cup C_{11} \cup C_{15}$	36
8	6	$C_0 \cup C_1 \cup C_3 \cup C_7 \cup C_{29} \cup C_{87} \cup C_{251}$	49
9	8	$C_0 \cup C_1 \cup C_3 \cup C_7 \cup C_{29} \cup C_{45} \cup C_{119} \cup C_{191} \cup C_{255}$	73
10	11	$C_0 \cup C_1 \cup C_3 \cup C_7 \cup C_{29} \cup C_{45} \cup C_{119} \cup C_{191} \cup C_{155} \cup C_{255} \cup C_{339}$	101

Table 5.4: Heuristics for choosing parameters t and L.

Counting the Linear Operations. From (5.29) and (5.30), we get $(2t-1)\cdot(|L|-1)+(t-1)$ as an upper-bound on the number of addition operations required to evaluate $P(x)$. This is because each of the $2t-1$ polynomials $p_i(x)$ and $q_i(x)$ in (5.30) have (at most) $|L|$ terms, and there are t summands in (5.30). From (5.34), we get:

$$(2t-1)\cdot(|L|-1)+(t-1) \leq 2\,t\,|L| \approx 2\cdot 2^n$$

Similarly, we get $(2t-1)\cdot|L| \approx 2\cdot 2^n$ as an estimate for the number of scalar multiplications. Since the squaring operations are used only to compute the list L, we need $|L|-\ell \leq |L| \approx \sqrt{n\cdot 2^n}$ many of them (cf. (5.37)).

Remark 5.8 The above count of the linear operations can be significantly reduced if the linear operations are replaced by table lookups as much as possible. Such an approach is particularly well suited for application in the higher-order masking scheme of [CGP+12], where we need to evaluate a given polynomial with many shares and that the processing of linear polynomials with shares is particularly straightforward. More specifically, we can write each $p_i(x)$ as a sum of $\mathbb{F}_2$-linear polynomials $p_{i,j}$, one for each cyclotomic class in the pre-computed set $\mathcal{S}$ (cf. (5.27), (5.28))[1]:

$$p_i(x) = \sum_{C_{\alpha_j}\in\mathcal{S}} p_{i,j}\left(x^{\alpha_j}\right).$$

The ℓ polynomials $p_{i,j}$ are $\mathbb{F}_2$-linear and hence are of the form $\sum_{k=0}^{n-1}\gamma_k\,x^{2^k}$. Similarly, the polynomials $q_i(x)$ can also be expressed in the above form. If we tabulate the values of each of the linear polynomials $p_{i,j}$ and $q_{i,j}$, then it suffices to evaluate x^{α_j} for each cyclotomic class $C_{\alpha_j}\in\mathcal{S}$ using only non-linear multiplications. Then the polynomials $p_{i,j}$ and $q_{i,j}$ can be evaluated by just table lookups, and then each of the $2t-1$ polynomials p_i and q_i can be eventually evaluated with $\ell-1$ additions each. Finally, we need $t-1$ more

[1] We thank Matthieu Rivain for this suggestion.

additions in the step (5.30). Hence, we need no scalar multiplications nor squarings using this table lookup technique. The total number of additions we need is

$$(2t-1)\cdot(\ell-1)+t-1 \approx \frac{2\cdot 2^n}{n}.$$

Note that this technique is not very effective for the evaluation method in Section 5.4.4.1 since nearly every linear polynomial that appears has at most two non-zero terms.

5.4 Application to various S-boxes

In this section, we apply the generic method described in Section 5.3.4, to several well known S-boxes. Using our new method, we reduce the number of non-linear multiplications required in each case, resulting in an improvement over the previously known techniques.

We stress that in our method for an n-bit S-box, the maximum number of non-linear multiplications required is invariant of the choice of the S-box when n is fixed. Hence, the number of non-linear multiplications obtained for a fixed n actually provides an upper bound on the masking complexity of an S-box of size n.

5.4.1 CLEFIA and Other 8-bit S-boxes

The CLEFIA block cipher has two 8-bit S-boxes [SSA+07]. Let us denote the S-box lookup table for either of the S-boxes as $\mathrm{S}_{\texttt{clefia}}$. We choose

$$L = C_0 \cup C_1 \cup C_3 \cup C_7 \cup C_{29} \cup C_{87} \cup C_{251}. \tag{5.40}$$

This implies that after choosing $t = 6$, and then 5 basis polynomials $q_i \xleftarrow{\$} \mathcal{P}(x^L)$ $(1 \le i \le 5)$, the following system of equations is constructed in $\mathbb{F}_{2^8}$:

$$\mathrm{S}_{\texttt{clefia}}[x_j] = \underbrace{\sum_{i=1}^{5} p_i(x_j)\cdot q_i(x_j)}_{Q} + p_6(x_j) \qquad j = 0,\ldots,255. \tag{5.41}$$

We have checked that for some random choice of the polynomials $q_i(x)$ the corresponding matrix A has full rank 256, and therefore we can determine the polynomials $p_i(x)$. Given the solution to the above system, the S-box evaluation is then the same as evaluating the polynomial $Q(x)+p_6(x)$. To evaluate all the monomials in $\{x, x^3, x^7, x^{29}, x^{87}, x^{251}\}$ we need 5 non-linear multiplications, implying that any monomial in x^L, any $q_i(x)$ (randomly chosen from $\mathcal{P}(x^L)$) and any $p_i(x)$ can all together be evaluated with 5 non-linear multiplications. Moreover the evaluation of $Q(x)$ requires 5 additional non-linear

multiplications. Therefore the total number of non-linear multiplications required for evaluating the S-box is 10.

Note that it requires at least 4 non-linear multiplications to evaluate the polynomials corresponding to the two S-boxes of CLEFIA by any method. This is because these two polynomials over $\mathbb{F}_{2^8}$ have degrees 252 (S-box S_0) and 254 (S-box S_1), and the result follows from Proposition 5.3.

Invariance. If we choose some other 8-bit S-box, then the matrix corresponding to the resulting system remains the same. Hence, we will still get a solution to the system for the same set of polynomials $q_i(x)$. This implies that we can use the same set of basis polynomials to obtain polynomials $p_i(x)$ for any other 8-bit S-box. Hence, for any S-box of size 8, the number of non-linear multiplications is at most 10.

5.4.2 PRESENT and Other 4-bit S-boxes

For the 4-bit S-box of PRESENT [BKL+07], we choose $t = 2$ and $L = C_0 \cup C_1 \cup C_3$. By selecting $q_1 \xleftarrow{\$} \mathcal{P}(x^L)$, we construct the following linear system of equations:

$$\mathrm{S}_{\mathtt{present}}[x_j] = p_1(x_j) \cdot q_1(x_j) + p_2(x_j) \tag{5.42}$$

The monomials used to construct $q_1(x)$, $q_2(x)$ are $\{x, x^2, x^4, x^8, x^3, x^6, x^{12}, x^9\}$. All of these monomials can be evaluated with a single non-linear multiplication and to evaluate $p_1(x) \cdot q_1(x)$ we need only one more non-linear multiplication. Hence, the PRESENT S-box evaluation requires 2 multiplications. As in the case of 8-bit S-boxes, this proves that with the same $q_1(x)$ any 4-bit S-box can be evaluated with 2 multiplications. Table 5.5 gives the corresponding polynomials for the PRESENT S-box.

The polynomial corresponding to the PRESENT S-box has degree 14 and hence, from Proposition 5.3, its masking complexity is at least 2 [RV13]. This implies that our evaluation method achieves optimal complexity for the PRESENT S-box.

5.4.3 (n, m)-bit S-box

We now consider S-boxes whose output size m is smaller than the input size n, as for the DES S-boxes with $n = 6$ and $m = 4$. We can view an (n, m)-bit S-box ($m < n$) as a mapping from $\mathbb{F}_{2^n}$ to $\mathbb{F}_{2^m}$. Given any such S-box table S,

	Basis Polynomial
q_1	$(a^3+a^2+1)\cdot x^{12}+(a^3+a^2+a+1)\cdot x^9+a^2\cdot x^8+x^6+$ $(a^3+a^2+a)\cdot x^4+x^2+(a^3+a)\cdot x+a$
	Solution to linear System
p_1	$(a^3+a)\cdot x^{12}+x^9+(a^3+a^2)\cdot x^8+(a^2+1)\cdot x^6+(a^3+a^2+$ $1)\cdot x^4+(a^3+a^2+a+1)\cdot x^3+(a^2+1)\cdot x^2+(a^2+1)\cdot x+a^2$
p_2	$(a^2+1)\cdot x^8+(a^3+a^2+1)\cdot x^6+(a+1)\cdot x^4+a\cdot x^3+x^2+$ $(a^3+1)\cdot x+a^2$

Table 5.5: Basis polynomial $q_1(x)$ for 4-bit S-boxes, and solutions $p_1(x), p_2(x)$ to PRESENT S-box. The irreducible polynomial is a^4+a+1 over $\mathbb{F}_2$.

we want to construct a system of linear equations

$$\mathsf{S}[x_j] = \underbrace{\sum_{i=1}^{t-1} p_i(x_j)\cdot q_i(x_j) + p_t(x_j)}_{G(x)} \tag{5.43}$$

Note that each $\mathsf{S}[x_j]$ is an element of the smaller field $\mathbb{F}_{2^m}$, but each $G(x_j)$ is an element in the larger field $\mathbb{F}_{2^n}$. One trivial way to remove this inconsistency is to consider $\mathsf{S}[x_j]$ as an element of the larger field $\mathbb{F}_{2^n}$, by padding the most significant bit of the S-box output with 0's. Then, we determine the polynomials $p_i(x)$ by solving the corresponding system $A\cdot\vec{c}=\mathsf{S}$, as described in Section 5.3.4.2. However intuitively this is not optimal, since we are creating an artificial constraint to be satisfied by the coefficients of the polynomials $p_i(x)$, namely that the $n-m$ most significant bits of $G(x)$ must be 0, while eventually these most significant bits will simply be discarded after the evaluation of $G(x)$, since to get $\mathsf{S}(x)$ we only keep the m least significant bits of $G(x)$.

Instead, we consider the representations of the unknown coefficients of the polynomials $p_i(x)$ in $\mathbb{F}_2$ instead of $\mathbb{F}_{2^n}$, and we transform the system of linear equations (5.43) over $\mathbb{F}_{2^n}$, into a system of linear equations over $\mathbb{F}_2$. By doing this, from each constraint $G(x_j)$, we generate m equations over $\mathbb{F}_2$, instead of one equation over $\mathbb{F}_{2^n}$. Note that each of these n equations will be an affine combination of the unknown bits of the coefficients of the polynomials $p_i(x)$. Only n of these equations are actually necessary, since the output of the S-box is of size m bits. By equating each of these equations to the corresponding output bit of the S-box, we get a transformed system of linear equations $B\cdot\vec{c}=\mathsf{S}$, where B is an $(m\cdot 2^n)\times(t\cdot|L|\cdot n)$ matrix over $\mathbb{F}_2$ and L is the set of elements from the chosen cyclotomic classes. By solving this transformed system over $\mathbb{F}_2$ we determine the polynomials $p_i(x)$.

5.4.4 DES S-boxes

5.4.4.1 Adapting the Divide-and-Conquer Method

Before we illustrate the above technique in Section 5.4.3 for the case of DES S-boxes, we will first describe our adaptation of the Divide-and-Conquer method from Section 5.3.4.1 for the DES S-boxes that appeared in [RV13]. This method requires only 7 non-linear multiplications compared to the Parity-Split method that requires 10 non-linear multiplications.

The DES block cipher has 8 $(6, 4)$-bit S-boxes [des93]. Let $P_{DES}(x) \in \mathbb{F}_{2^6}[x]$ be the Lagrange interpolation polynomial corresponding to a DES S-box. Here the 4-bit output of a DES S-box is identified as a 6-bit output with two leading zeroes, and hence these bit strings are naturally identified with the elements of $\mathbb{F}_{2^6}$. Note that for all the DES S-boxes, $\deg(P_{DES}(x)) = 62$. Write

$$P_{DES}(x) = q(x) \cdot x^{36} + R(x),$$

where $\deg(R) \leq 35$ and $\deg(q) = 26$. Then divide the polynomial $R(x) - x^{27}$ by $q(x)$:

$$R(x) - x^{27} = c(x) \cdot q(x) + s(x),$$

where $\deg(c) \leq 9$ and $\deg(s) \leq 25$, which gives

$$P_{DES}(x) = \big(x^{36} + c(x)\big) \cdot q(x) + x^{27} + s(x).$$

Next decompose the polynomials $q(x)$ and $x^{27} + s(x)$ in a similar way but, instead, dividing first by x^{18}, and then using x^9 as the "correction term". One gets

$$q(x) = (x^{18} + c_1(x)) \cdot q_1(x) + x^9 + s_1(x),$$
$$x^{27} + s(x) = (x^{18} + c_2(x)) \cdot q_2(x) + x^9 + s_2(x)$$

where $\deg(q_1) = 8$, $\deg(c_1) \leq 9$, $\deg(s_1) \leq 7$, $\deg(q_2) = 9$, $\deg(c_2) \leq 8$, and $\deg(s_2) \leq 8$. Finally,

$$\begin{aligned} P_{DES}(x) =& (x^{36} + c(x)) \cdot \Big(((x^{18} + c_1(x)) \cdot q_1(x)) + (x^9 + s_1(x))\Big) \\ &+ \Big((x^{18} + c_2(x)) \cdot q_2(x) + (x^9 + s_2(x))\Big). \end{aligned} \tag{5.44}$$

The monomials x, x^2, x^3, x^4, x^5, x^6, x^7, x^8, x^9, x^{18}, x^{36} are first evaluated using 4 non-linear multiplications. Namely a non-linear multiplication is required for each of the monomials x^3, x^5, x^7 and x^9; the rest of the monomials can be evaluated using linear squarings only. Each of the individual polynomials in the above expression such as $x^{36} + c(x)$, $x^{18} + c_1(x)$, $q_1(x)$, and so on, can then be evaluated for free, that is without further non-linear multiplications. To evaluate $P_{DES}(x)$ from (5.44), 3 more non-linear multiplications are needed, and hence totally 7 non-linear multiplications are sufficient to evaluate a DES S-box.

5.4.4.2 Improved Method

From Table 5.3, we can see that by using our generic method over $\mathbb{F}_{2^6}$ we can perform the evaluation with 5 non-linear multiplications. Below we show that by working over $\mathbb{F}_2$ as explained in Section 5.4.3, only 4 non-linear multiplications are required.

We choose $L = C_0 \cup C_1 \cup C_3 \cup C_7$, $t = 3$, and $q_1(x), q_2(x) \xleftarrow{\$} \mathcal{P}(x^L)$. Then using our method we transform the following linear system of equations

$$\mathrm{S}_{\mathtt{des}}[x_j] = \underbrace{\sum_{i=1}^{2} p_i(x_j) \cdot q_i(x_j)}_{Q(x)} + p_3(x_j) \tag{5.45}$$

to a system over $\mathbb{F}_2$. That is, instead of embedding $\mathrm{S}_{\mathtt{des}}$ into $\mathbb{F}_{2^6}$, we write the system of equations over $\mathbb{F}_2$. This can be done by considering the binary representation of x^α evaluated at any given value in $\mathbb{F}_{2^6}$. This will give 6 equations over $\mathbb{F}_2$ for each equation $Q(x_j) + p_3(x_j)$. Out of these 6 equations only 4 will be necessary since the output of DES S-box has 4-bit values. By solving this new system of linear equations over $\mathbb{F}_2$ we can determine $p_i(x)$ for each i.

The number of multiplications required to evaluate $q_1(x), q_2(x)$ is 2, and $Q(x)$ can be evaluated with 2 additional multiplications. Hence, the total number of non-linear multiplications required is only 4. In Section 5.4.5 we give an example of basis polynomials $q_1(x)$, $q_2(x)$ for DES and the solution polynomials $p_i(x)$ corresponding to the system of linear equations for the first DES S-box S_1.

As previously, once we obtain a full rank matrix for a set of randomly fixed $q_1(x), q_2(x)$, for any other $(6, 4)$-bit S-box we can use this basis to find the corresponding polynomials $p_i(x)$, since the matrix A is independent from the S-box. Hence we can conclude that the masking complexity of any $(6, 4)$-bit S-box is at most 4.

5.4.5 Evaluation Polynomials for DES S-boxes

In Table 5.6 we give an example of basis polynomials $q_1(x)$, $q_2(x)$ for DES and Table 5.7 shows the solution polynomials $p_i(x)$ corresponding to the system of linear equations for the first DES S-box S_1.

5.4.6 Implementation Results: DES

We have performed a software implementation of the CGPQR countermeasure [CGP+12] for DES that incorporates our new polynomial evaluation technique

Basis Polynomials	
q_1	$(a^5+a^4+1)\cdot x^{56}+(a^5+1)\cdot x^{49}+(a^2+a)\cdot x^{48}+(a^4+a^3)\cdot x^{35}+(a^5+a^4+a^2)\cdot x^{33}+(a^5+a+1)\cdot x^{32}+(a^3+a)\cdot x^{28}+a^2\cdot x^{24}+(a^5+1)\cdot x^{16}+(a^4+a+1)\cdot x^{14}+x^{12}+(a^4+a^3+a^2+1)\cdot x^8+(a^5+a^3+a^2+a+1)\cdot x^7+(a^5+a^4+a^3+a^2+1)\cdot x^6+(a^5+a^4+a^3+1)\cdot x^4+(a^5+a^2+a+1)\cdot x^3+(a^3+a^2+a)\cdot x^2+(a^4+a^2+a+1)\cdot x+a^5+a^4+a^3+a^2+a$
q_2	$(a+1)\cdot x^{56}+(a^5+1)\cdot x^{49}+(a+1)\cdot x^{48}+a\cdot x^{35}+(a+1)\cdot x^{33}+(a^4+a^3+a+1)\cdot x^{32}+(a^3+a^2+a)\cdot x^{28}+(a^5+a^3+a+1)\cdot x^{24}+(a^3+1)\cdot x^{16}+(a^4+a^2+1)\cdot x^{14}+(a+1)\cdot x^{12}+(a^5+a^4+1)\cdot x^8+(a^5+a^4+a^3+a+1)\cdot x^7+(a^5+a^4+a^3)\cdot x^6+(a+1)\cdot x^4+(a^5+a^3+a^2+a)\cdot x^2+a\cdot x+a^5+a^4+a^3+a^2+1$

Table 5.6: Basis polynomials q_1, q_2 obtained from $\mathcal{P}(x^L)$, for DES.

Solution to linear system	
p_1	$(a^5+a^4+a^3+a^2+1)\cdot x^{56}+(a^5+a^2+1)\cdot x^{49}+a^4\cdot x^{48}+(a^4+a^3+a)\cdot x^{35}+(a^5+a^4+a^2)\cdot x^{33}+(a^5+1)\cdot x^{32}+a\cdot x^{28}+(a^4+a^2)\cdot x^{24}+(a^5+a)\cdot x^{16}+(a^5+a^2)\cdot x^{14}+(a^5+a+1)\cdot x^{12}+(a^5+a^4+a^3+a)\cdot x^8+(a^5+a^4+a^3+a)\cdot x^7+(a^5+a^4+a^3)\cdot x^6+(a^2+a+1)\cdot x^4+(a^5+a^4+a)\cdot x^2+(a^5+a^4+1)\cdot x+a^4+a^3+a^2$
p_2	$(a^5+a^2)\cdot x^{49}+(a^3+1)\cdot x^{48}+(a^5+a^3+a+1)\cdot x^{35}+(a^4+a^2+1)\cdot x^{33}+(a^5+a^4+1)\cdot x^{32}+(a^5+a^4+a^3+a+1)\cdot x^{28}+(a^3+a^2)\cdot x^{24}+(a^2+a+1)\cdot x^{16}+(a^5+a^4+a^3)\cdot x^{14}+(a^4+a^3+a+1)\cdot x^{12}+(a^4+a^3)\cdot x^8+(a^5+a)\cdot x^7+(a^5+a^4)\cdot x^6+(a^5+a^4+a^3+a^2+a+1)\cdot x^4+(a^5+a^4+a)\cdot x^3+(a^5+a^3+a+1)\cdot x^2+(a^5+a)\cdot x+a^5+a^4+a^2+a$
p_3	$a\cdot x^7+a\cdot x^6+(a^4+a+1)\cdot x^4+(a^5+a^2+a)\cdot x^3+(a^5+a^4+a+1)\cdot x^2+(a^4+a^2)\cdot x$

Table 5.7: Solution to the system of linear equations for DES S-box (S_1). The irreducible polynomial is a^6+a+1 over $\mathbb{F}_2$.

Table 5.8: Comparison of secure implementations of DES.

Method	t'	n'	Rand $\times 10^3$	Mem (bytes)	Time (ms)	PF
Unprotected					0.008	1
CGPQR+RV	1	3	2752	72	0.552	69
Table Recomputation	1	3	8512	423	0.279	34
CGPQR+CRV	1	3	2368	40	0.166	20
CGPQR+RV	2	5	9152	118	0.966	120
Table Recomputation	2	5	33472	691	0.612	76
CGPQR+CRV	2	5	7872	64	0.336	42
CGPQR+RV	3	7	19200	164	1.507	188
Table Recomputation	3	7	74880	959	1.091	136
CGPQR+CRV	3	7	16512	88	0.591	73
CGPQR+RV	4	9	32896	210	2.167	270
Table Recomputation	4	9	132736	1227	1.696	212
CGPQR+CRV	4	9	28288	112	0.905	113

from Section 5.4.3 (referred to as the CGPQR+CRV method in Table 5.8) requiring only 4 non-linear multiplications. We have implemented this in C on a Dell Latitude 13 notebook running Ubuntu 12.04 Linux. The processor is Intel Core 2 Duo (32-bit architecture) running at 1.3 GHz. Our implementation is based on the source code available from [Cor13]. The present implementation is also publicly available at [Cor13]. We have used the technique of tabulating linear polynomials from Remark 5.8 in the implementation of our polynomial evaluation method. Note that these tables corresponding to the linear polynomials need to be stored only in the ROM.

In Table 5.8, we have compared the above timing results with that of the CGPQR countermeasure implemented with the technique from Section 5.4.4.1 (referred to as the CGPQR+RV method) that requires 7 non-linear multiplications. We have also made a comparison with the higher-order table recomputation method of Coron [Cor14]. In Table 5.8, the parameter t' refers to the order of security and n' refers to the number of shares in the full security model of [ISW03]. Note the relation $n' = 2t' + 1$. The (RAM) memory requirement (in bytes) is provided only for the S-box computations and the overall execution time for a DES encryption is in milliseconds. The penalty factor (PF) gives the ratio of the execution time of a given method to that of an unprotected implementation. The number of calls to the random number generator is 1000 times that of the reported quantity.

5.5 Conclusion and Future Directions

In this chapter, we studied various methods to evaluate polynomials over $\mathbb{F}_{2^n}$ that reduces the number of non-linear multiplications required. We proposed an asymptotically optimal (but heuristic) algorithm for this problem. As a consequence of our results, in practice, we can significantly improve the efficiency of generic higher-order masking scheme of Carlet et al.

It will be interesting to explore further the cost model of Grosso et al. [GPS14]. In [GPS14], non-linear multiplications are classified into two types: type-II and type-III. A type-II non-linear multiplication is of the form $y \times g(y)$, where $g()$ is an $\mathbb{F}_2$-affine function, whereas type-III non-linear multiplications are the complement of type-II operations. Type-II operations can be implemented twice as efficiently compared to type-III operations for medium-sized S-boxes. In fact, we have observed that by suitably adapting our evaluation method, we can evaluate DES S-boxes with only 1 type-III operation and 3 type-III operations. An open question is whether can we evaluate DES S-boxes with only 5 type-III operations?

An interesting future direction is to further improve our method from Section 5.3.4.2 by obtaining smaller decompositions (i.e., having smaller value for t) without increasing the precomputed set of monomials. One possible approach is to use Gröbner basis, though the magnitude of the resulting multivariate quadratic system of equations seems challenging for current techniques.

Another possible application of the non-linear cost model we have used in this chapter is in the design of fully homomorphic encryption schemes [Gen09]. For efficiency reasons, it is desirable to express circuits of additions and multiplications over $\mathbb{F}_2$that we wish to homomorphically evaluate as low depth circuits w.r.t. to the multiplication operations [GHS12]. Eventually, the techniques developed in this chapter may find greater use in such designs.

Our results for binary finite fields seem to readily extend to finite fields of general characteristic. It will be interesting to find practical applications of the non-linear cost model for the case of finite fields of general characteristic.

Part III

Simulatable Leakage Assumption: Efficient Symmetric-Key Constructions

Chapter 6

Leakage-Resilient Symmetric-Key Authentication and Encryption

Because of their efficiency and usability on a wide range of platforms, leakage-resilient cryptosystems based on symmetric-key primitives, such as block ciphers, are particularly attractive. So far, the literature has mostly focused on the design of leakage-resilient pseudorandom objects, e.g., PRGs, PRFs, PRPs. In this chapter, we consider the complementary and practically important problem of designing secure authentication and encryption schemes. We follow a pragmatic approach based on the advantages and limitations of existing leakage-resilient pseudorandom objects, and rely on the arguably necessary, yet minimal, use of a leak-free component for this purpose. The latter can typically be instantiated with a block cipher implementation protected by traditional masking-based countermeasures, and we investigate how to combine it with the more intensive use of a much more efficient (less protected) block cipher implementation.

Based on these premises, we propose and analyze new constructions of leakage-resilient MAC and encryption schemes, which allow fixing security and efficiency drawbacks of previous proposals in this direction. For encryption, we additionally provide a detailed discussion of why previously proposed indistinguishability-based security definitions cannot capture actual side-channel attacks, and suggest a relaxed and more realistic way to quantify leakage-resilience in this case, by reducing the security of many iterations of the primitive to the security of a single iteration, independent of the security notion guaranteed by this single iteration (that remains hard to define).

Contents

6.1 Introduction

Symmetric cryptographic primitives such as block ciphers are of utmost importance and in general, they are considered as the workhorses of modern cryptography [KR11]. Because of their low cost and efficiency on a wide range of platforms, they are also a target of choice for physical attacks. Unfortunately, their lack of mathematical structure makes them particularly challenging to protect. Taking the example of side-channel attacks, probably the most investigated countermeasure is masking [CJRR99] (a.k.a. secret sharing [ISW03]). But it implies overheads that are at least quadratic in the number of shares used [GSF14], and its secure implementation is far from trivial, i.e., hardware engineers have to ensure that the leakage of each shares is independent of each other, which may lead to further constraints [NRS11].

So there is a need for leakage-resilient symmetric primitives, which by design is inherently more secure against such physical attacks. So far, leakage-resilient symmetric cryptography has mostly focused on PRGs (a.k.a. stream ciphers) [DP08, FPS12, PSP$^+$08, Pie09, SPY13, SPY$^+$10, YS13, YSPY10], PRFs and PRPs [DP10, FPS12, SPY$^+$10, YS13]. By contrast, much less work has been carried out on the exploitation of these leakage-resilient primitives in the context of standard cryptographic tasks such as authentication and encryption. Our goal in this chapter is therefore to clarify how and when to use leakage-resilience in these cases, and for what kind of formal security guarantees.

6.1.1 Preliminaries

Our starting point for dealing with this problem is a recent work of Belaïd, Grosso and Standaert [BGS15] that shows that concretely, the security improvements brought by leakage-resilience highly depend on whether the underlying primitive is stateful (like PRGs, typically) or stateless (like PRFs and PRPs, typically). By stateful, we mean that the implementation of the primitive has to maintain a state (typically a key) between its consecutive iterations, which implies that different parties involved in the use of this primitive have to be synchronized. That is, despite proofs for both types of primitives being essentially based on the same assumptions, namely, the leakage per iteration has to be limited in some sense, ensuring this condition in practice is significantly more difficult in the case of stateless primitives than in the case of stateful ones.

In the case of PRGs and stream ciphers, leakage-resilient designs limit the number of measurements that an adversary can obtain per iteration. By contrast, for PRFs and PRPs, they only limit the number of plaintexts for which measurements can be obtained, which still allows the adversary to measure the same plaintext an arbitrary number of times, hence to reduce the noise. Therefore, implementations of leakage-resilient PRGs and stream ciphers (mostly) lead to concrete security against side-channel key recovery attacks at a lower cost than countermeasures like masking. By contrast, implementations of leakage-resilient PRFs and PRPs (mostly) lead to lower concrete security levels than standard PRFs and PRPs protected with such countermeasures.

As a result, if we want to stick with constructions based on standard block ciphers for efficiency and usability reasons, there seems to be little hope to have a secure MAC or encryption scheme without further assumption. Indeed, stateless primitives are usually important ingredients of such schemes, and without properties such as a homomorphic structure, block cipher re-use will eventually leak the key in full, as just explained. In this respect, a natural direction to investigate is to assume that we will need a leak-free component, i.e., a leak-free block cipher in our case. Admittedly, this leak-free component will be much slower than an unprotected block cipher implementation, as it could be based on a combination of masking and other countermeasures, in fact, it could also be based on a primitive enjoying some exploitable homomorphic structure.

So our goal will be to make minimal use of this component, typically, one call per message, independently of the message, and to combine it with a faster implementation of block cipher in order to get a scheme that would still provide good protection against side-channel attacks (or at least, as good as we can hope), but would also be much more efficient than if we had to use the leak-free component only, or solutions that only exploit the mathematical structure of asymmetric cryptographic primitives such as [KP10a] (also Chapter 2) for encryption, and [MOS13], Chapters 3 and 4 for authentication.

6.1.2 Leakage Model

We consider the continuous leakage model since it is the only one capturing actual side-channel attacks. Indeed, if a system is used for a sufficiently long period of time, the amount of leakage observed by an attacker may exceed any apriori determined leakage bound. In this context, we capture the limited informativeness of actual leakages with the recently introduced "simulatable leakage" framework [SPY13]. We are aware of the ongoing discussion about how to implement block ciphers ensuring this empirically verifiable assumption [GMO+14, PSMD14]. Yet, and as argued in these different papers, it remains the most realistic assumption to reason about leakage we currently have. Besides, and more importantly, we believe our main contribution is the general discussion of leakage-resilient MAC and encryption, as well as the proposal of new efficient constructions minimizing the need of leak-freeness. That is, we use a leakage model, here, the simulatability framework, to reason formally about our constructions and make sure that they are theoretically well founded. But we also hope that they will be helpful in practice, for cryptographic engineers.

6.1.3 Our Contribution

First, we follow this goal of minimizing the need of leak-freeness for two important symmetric cryptographic functionalities, namely authentication and encryption. Second, we clarify and fix two important shortcomings in previously published approaches to these functionalities.

For leakage-resilient MACs, the only existing work based on symmetric primitives is the one by Schipper [Sch10]. The basic idea is simple: take a leakage-resilient PRG and use it to generate keys for a one-time MAC. While this is indeed a stateful primitive, the main problem in this scheme is that the use of the key is limited per message, not per message block. This means that for long messages, and depending on the one-time MAC that is used, the adversary can observe a large number of leaking operations exploiting the same key. For instance, CBC-MAC would be problematic.

One partial solution considered in Schipper's thesis is to use a MAC based on a leakage-resilient PRF. But this has a higher implementation cost, as noticed in [MOS13], and faces the previously discussed problem of stateless primitives. In order to improve this situation, we first propose a new (stateful) leakage-resilient MAC that limits the use of leak-free component to a single block of IV, which can therefore be pre-computed, and is efficient for large messages, i.e., it requires a single block cipher execution per message block. We then propose a further simplification of this scheme based on a hash and MAC paradigm. Along these lines, we also put forward that certain standard MACs are better suited for leakage-resilience than others, for e.g., HMAC is better than CBC-MAC in this respect.

For encryption, the literature based on symmetric primitives is also sparse. To the best of our knowledge, the work by Abdalla et al. is the only one to address this question [ABF13]. Here, the problem is more general and definitional. That is, a central issue in all the leakage-resilient encryption schemes proposed so far is that they need to exclude the leakage during the challenge queries, or restrict themselves to settings where an encryption is assumed to not leak any single bit of the plaintext that is encrypted, for e.g., consider only key leakage. In fact, this is also true for public-key encryption schemes: see, e.g. [NS09] for an early proposal in this direction and [HLWW13] for a more recent one. On the one hand, this seems unavoidable: indeed a single bit of leakage on the plaintext trivially breaks the semantic security game. On the other hand, we argue that excluding challenge leakages is artificial and does not capture the actual adversarial scenario of leaking devices, at least in the context of side-channel attacks based on power consumption and electromagnetic radiation that we consider in this chapter, but we believe that in general as well, this is the case.

Hence, we propose an alternative way to model the security in front of leakage where we do not try to enforce traditional security notions with a negligible advantage. We rather show that the security of multiple iterations reduces to the security of a single iteration. That is, we show that whatever the adversary is able to do against multiple iterations of our encryption scheme is also possible against a single iteration of this scheme. We believe this approach is more realistic since it does not give users the (illusive) feeling that semantic or any indistinguishability-based security can be obtained for encryption schemes with leakage. By contrast, we provide an efficient solution for which the designer is guaranteed that the security of the full construction reduces to the security of a single block, whatever security he is able to achieve for this single block.

6.2 Leakage-Resilient Message Authentication Codes

6.2.1 Security Definition

Let us recollect the standard definition of a Message Authentication Code (MAC). A MAC is a tuple of three polynomial time algorithms $\mathsf{MAC} = (\mathsf{KeyGen}, \mathsf{Mac}, \mathsf{Vrfy})$ defined as follows:

- the **key generation** algorithm $\mathsf{KeyGen}(1^n)$ takes as input the security parameter n and generates a shared (master) secret key k to be used by both the tag generator and the verifier,
- the **MAC generation** algorithm $\mathsf{Mac}(m, k)$ takes as input the message m and the secret key k (also possibly some randomness), and then outputs a tag τ.

– the **tag verification** algorithm $\mathsf{Vrfy}(m, \tau, k)$ takes as input a message m, the corresponding tag τ and the secret key k. The algorithm outputs 1 (*accept*) if the tag is valid for the message, else it outputs 0 (*reject*).

We require the usual *correctness property* to be satisfied by the MAC, i.e. $\mathsf{Vrfy}(m, \mathsf{Mac}(m, k), k) \rightarrow 1$ for every message m and key k in the range of KeyGen. Informally, the MAC is said to be *existentially unforgeable in the presence of (session) key leakage during tag generation* (in short, LR-MAC) if the adversary is unsuccessful in the following security game. The adversary can obtain tags corresponding to the messages of its choice. Every time a tag is computed, a session key is used, which is derived using the shared master secret key k and some fresh randomness. The adversary will not be able to see the leakages during the session key generation due to the leak-free component. However, it will be able to obtain the leakages corresponding to the session key during the tag generation. The adversary will also have access to the verification oracle, but it will not be able to get the corresponding leakages during verification that corresponds to the fact that we only secure the tag generation against side-channel attacks, not the tag verification.

Note that as mentioned in the introduction, preventing side-channel attacks during the tag verification will most likely be more challenging. Indeed, if the adversary can observe the leakage during verification, it should be able to fully recover the key by re-using it many times in the verification phase. Besides, this problem is not specific to symmetric primitives. A similar (yet relaxed) restriction is also made in [MOS13] for a construction of a leakage-resilient MAC based on pairings. In this case, the adversary gets the leakage during verification, but only once, for a given message and randomness pair. Otherwise, the adversary will be able to leak a correct tag bit-by-bit by repeatedly accessing the verification oracle with incorrect tags against the same message. The goal of the adversary is to output a valid forgery on a message for which it has not previously obtained a corresponding tag.

Formally, we consider the following experiment $\mathsf{Forge}^{\mathsf{euf-cma}}_{\mathcal{A}^{\mathsf{L}},\mathsf{MAC}}(n)$:

$\mathsf{Forge}^{\mathsf{euf-cma}}_{\mathcal{A}^{\mathsf{L}},\mathsf{MAC}}(n)$
$k \leftarrow \mathsf{KeyGen}(1^n)$
$\mathcal{F} \leftarrow \emptyset$
$(m, \tau) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{ML}}(\cdot),\mathcal{O}^{\mathsf{V}}(\cdot)}(1^n)$
If $m \in \mathcal{F}$, then return $b := 0$
$b \leftarrow \mathsf{Vrfy}(m, \tau, k)$

Oracle $\mathcal{O}^{\mathsf{ML}}(m)$
$\tau \leftarrow \mathsf{Mac}(m, k) \rightsquigarrow \ell$
$\mathcal{F} \leftarrow \mathcal{F} \cup m$
Return τ, ℓ

Oracle $\mathcal{O}^{\mathsf{V}}(m, \tau)$
Return $\mathsf{Vrfy}(m, \tau, k)$

Here, ℓ represents the leakage that occurs when performing the tag computation and specifically excludes leakage during the session key generation. More precisely, the leakage depends on the message and the session key, which is derived from the master secret key and some fresh randomness.

Definition 6.1. [LR-MAC] MAC *is said to be* (q, s, t, ϵ) *secure LR-MAC if for any adversary* $\mathcal{A}$ *running in time at most* t*, making at most* q *MAC queries and additionally at most* s *queries to the leakage oracle, its advantage in the experiment* $\mathsf{Forge}^{\mathsf{euf-cma}}_{\mathcal{A}^{\mathsf{L}},\mathsf{MAC}}(n)$ *is at most* ϵ*. That is,*

$$\Pr[\mathsf{Forge}^{\mathsf{euf-cma}}_{\mathcal{A}^{\mathsf{L}},\mathsf{MAC}}(n) = 1] \leq \epsilon.$$

Remark 6.1 The notion of *Strongly-Unforgeable LR-MAC* is a stronger security notion than that in Definition 6.1. This distinction is analogous to the difference between strong unforgeability and basic unforgeability notions for MAC in the traditional setting (without leakage). In the case of strongly-unforgeable LR-MAC, it suffices for the adversary to output a valid message-tag pair distinct from those pairs it received in its interaction. Hence, it is not necessarily required to output forgery on a distinct message. Otherwise, the security game remains the same as that for LR-MAC.

6.2.2 Why CBC-MAC is not Leakage-Resilient?

To motivate our following investigations, we start with a brief explanation why standard MAC constructions such as CBC-MAC, are not leakage-resilient by default. For this purpose, just look at the informal description of CBC-MAC in Figure 6.1. Here, the master key k is used in every iteration of the MAC (and kept constant among messages). So we are exactly in the scenario where a standard side-channel key recovery attack is the most devastating. As a result, a natural suggestion for improving the situation would be to combine CBC-MAC with a leakage-resilient stream cipher so that every message block would be processed with a different key. Yet, this would imply three block cipher executions per message block. In the following section, we show that a three times more efficient solution can be obtained.

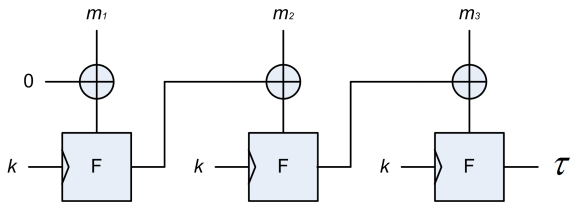

Figure 6.1: CBC-MAC.

6.2.3 Leakage-Resilient Tag Generation with Re-Keying

We next present a plausible construction of LR-MAC that is a variant of the standard CBC-MAC. The scheme $\mathsf{MAC}_1 = (\mathsf{KeyGen}_1, \mathsf{Mac}_1, \mathsf{Vrfy}_1)$, depicted in Figure 6.2 and described below, is a fixed length MAC that takes as input l blocks of messages ($l \geq 1$), each block being n-bit long. The construction

requires a pseudorandom function F that we will typically instantiate with a block cipher.

Description of MAC_1:

- $\mathsf{KeyGen}_\mathbf{1}(1^n)$: Choose a shared master secret key $k \xleftarrow{\$} \{0,1\}^n$.
- $\mathsf{Mac}_\mathbf{1}(m,k)$: Parse $m = \langle m_1, m_2, \ldots, m_l \rangle$. Choose $IV \xleftarrow{\$} \{0,1\}^n$. Compute the session key $k' := k_1 = \mathsf{F}_k(IV)$.
  - for $j = 2, \ldots l+1$: compute $k_j = \mathsf{F}_{k_{j-1}}(m_{j-1})$.

  Return $\tau = (IV, k_{l+1})$.
- $\mathsf{Vrfy}_\mathbf{1}(m,\tau,k)$: Parse $\tau = (IV, tg)$. Compute $\tau' \leftarrow \mathsf{Mac}_1(m,k,IV)$.
  - If $\tau' \stackrel{?}{=} \tau$, then **return** 1 (correct), else **return** 0 (incorrect).

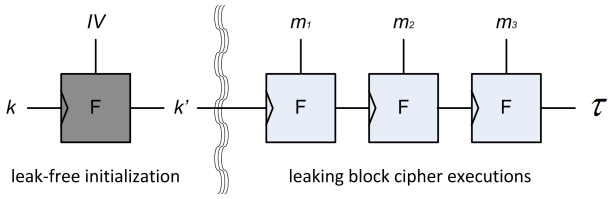

Figure 6.2: Re-keying MAC.

Compared to CBC-MAC, one can directly see that this scheme brings improved leakage-resilience, since a new session key is used for every new message. Compared to the LR-MAC of Schipper in [Sch10], we have the additional advantage that the key evolves for every message block, which allows us to state our requirements on the leakage for a single iteration of the scheme. We also exploit the block cipher quite efficiently since this new stateful construction essentially requires an execution of F per message block. Eventually, we require a very minimum use of the leak-free component (depicted in dark grey on the figure): it is only needed to encrypt a random IV under the master key k.

Remark 6.2 In the MAC_1 construction above, a random IV is chosen to compute every new tag on a message m. Nevertheless, the security of the construction will not be affected even if we choose the IVs arbitrarily, as long as they are distinct (cf. proof of Theorem 6.1). Hence, for instance, we could use a counter mode (i.e. start with $IV = 0$, and then successively increment it), but this would require the MAC to maintain a state.

We now prove the LR-MAC security of our MAC_1 construction based on the pseudorandomness and the simulatable leakage assumption of the block cipher F, assuming that the block cipher implementation F used on IV to compute the session key k' is leak-free. We first recall these properties.

Definition 6.2. [Pseudorandom Function (PRF) [SPY13, Definition 2]] *A block cipher* $\mathsf{F} : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ *is a* (s, t, ϵ_{prf}) *PRF in the presence of leakage function* L *if, for every* (s,t)*-bounded adversary* $\mathcal{A}^{\mathsf{L}(.,.)}$*, we have:*

$$|\Pr[\mathcal{A}^{\mathsf{L}(.,.),\mathsf{F}_K(\cdot)} = 1] - \Pr[\mathcal{A}^{\mathsf{L}(.,.),\mathsf{R}(\cdot)} = 1]| \leq \epsilon_{prf},$$

where $K \xleftarrow{\$} \{0,1\}^n$ *and* R *is a random function.*

As discussed in [SPY13], this definition would be exactly equivalent to the standard notion of PRF if the leakage function was polynomial time. However, it remains an open problem to determine the exact complexity of physical functions. Hence, the definition is augmented in order to allow the adversary to access a leakage oracle independent of the PRF challenge (which captures the fact that in theory, nothing precludes this leakage function to do some cryptanalytic work).

Next, the q-simulatable leakage assumption is defined via the following game.

Game q-$\mathsf{sim}(\mathcal{A}, \mathsf{F}, \mathsf{L}, \mathcal{S})$ [SPY13, Section 2.1].

The challenger selects two random keys $k, k^* \xleftarrow{\$} \{0,1\}^n$ *and a random bit* $b \xleftarrow{\$} \{0,1\}$*. The output of the game is a bit* b' *computed by* $\mathcal{A}^{\mathsf{L}}$ *based on the challenger responses to a total of at most* q *adversarial queries of the following type:*

Query	Response if $b = 0$	Response if $b = 1$
$\mathsf{Enc}(x)$	$\mathsf{F}_k(x)$, $\mathsf{L}(k, x)$	$\mathsf{F}_k(x)$, $\mathcal{S}^{\mathsf{L}}(k^*, x, \mathsf{F}_k(x))$

and one query of the following type:

Query	Response if $b = 0$	Response if $b = 1$
$\mathsf{Gen}(z, x)$	$\mathcal{S}^{\mathsf{L}}(z, x, k)$	$\mathcal{S}^{\mathsf{L}}(z, x, k^*)$

It directly leads to the following definition of a block cipher implementation with q-simulatable leakages.

Definition 6.3. [q-simulatable leakages [SPY13, Definition 1]] *Let* F *be a block cipher having leakage function* L*. Then* F *is said to have* $(s_{\mathcal{S}}, t_{\mathcal{S}}, s_{\mathcal{A}}, t_{\mathcal{A}}, \epsilon_{q\text{-}sim})$ *q-simulatable leakages if there is an* $(s_{\mathcal{S}}, t_{\mathcal{S}})$*-bounded simulator* $\mathcal{S}^{\mathsf{L}}$ *such that, for every* $(s_{\mathcal{S}}, t_{\mathcal{S}})$*-bounded adversary* $\mathcal{A}^{\mathsf{L}}$*, we have:*

$$|\Pr[q\text{-}\mathsf{sim}(\mathcal{A}, \mathsf{F}, \mathsf{L}, \mathcal{S}^{\mathsf{L}}, 1) = 1] - \Pr[q\text{-}\mathsf{sim}(\mathcal{A}, \mathsf{F}, \mathsf{L}, \mathcal{S}^{\mathsf{L}}, 0) = 1]| \leq \epsilon_{q\text{-}sim}.$$

Eventually, the following lemma is a consequence of Definition 6.2 and Definition 6.3 (for $2-$ simulatable leakages) [SPY13].

Lemma 6.1. [2-simulatable ideal execution [SPY13, Lemma 1]] *Let* $\mathsf{F} : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ *be a* (s, t, ϵ_{prf}) *PRF in the presence of leakage function* L *having* $(s_{\mathcal{S}}, t_{\mathcal{S}}, s, t, \epsilon_{2\text{-}sim})$ *2-simulatable leakages, and let* $\mathcal{S}^{\mathsf{L}}$ *be an appropriate*

$(s_{\mathcal{S}}, t_{\mathcal{S}})$-bounded leakage simulator. Then, for every $k^-, p_0, p_1, z \in \{0,1\}^n$ and every $(s - 3s_{\mathcal{S}}\ , t - \max(t_{prf}, t_{sim}))$-bounded distinguisher $\mathcal{D}^{\mathsf{L}}$, the following holds:

$$\Big|\Pr\left[\mathcal{D}^{\mathsf{L}}\left(y^+, k^+, \mathsf{L}(k', p_0), \mathsf{L}(k', p_1), \mathcal{S}^{\mathsf{L}}(k^-, z, k')\right) = 1\right] -$$
$$\Pr\left[\mathcal{D}^{\mathsf{L}}\left(y^{+*}, k^{+*}, \mathcal{S}^{\mathsf{L}}(k', p_0, y^{+*}), \mathcal{S}^{\mathsf{L}}(k', p_1, k^{+*}), \mathcal{S}^{\mathsf{L}}(k^-, z, k')\right) = 1\right]\Big| \leq \epsilon_{prf} + \epsilon_{2\text{-}sim},$$

with $k', y^{+*}, k^{+*} \xleftarrow{\$} \{0,1\}^n$, $y^+ = \mathsf{F}(k', p_0)$, $k^+ = \mathsf{F}(k', p_1)$, t_{prf} being equal to $3t_{\mathcal{S}}$ augmented with the time needed to make 2 oracle queries to the PRF challenger and select a uniformly random key in $\{0,1\}^n$, and t_{sim} being the time needed to relay the content of two Enc *and one* Gen *queries from and to a q-sim challenger.*

Remark 6.3 We note that the output of the two Enc and the Gen queries in Lemma 6.1 can be obtained adaptively. More precisely, let $\langle d_1, d_2, d_3, d_4, d_5\rangle$ denote the input received by $\mathcal{D}^{\mathsf{L}}$. The above indistinguishability result holds even if $\mathcal{D}^{\mathsf{L}}$ adaptively obtains the input as $\langle d_1, d_3\rangle$, $\langle d_2, d_4\rangle$, and d_5, in any order of its choice. This observation will be useful in the security analysis of MAC_1.

Remark 6.4 Note that besides the previously mentioned simulatability, we need to assume that blocks need to leak independently of each other. This actually corresponds to the "only computation leaks" assumption (or "independent leakage" assumption) that is anyway required for any proof of leakage-resilience to hold. In the present case, we believe that it is reasonable to have it satisfied in practice, since we need it at the macroscopic level of fairly large blocks. That is, as for [DP08] and follow up works, it seems unlikely that our construction will be broken because of small deviations from this assumption (which can possibly be reduced at the hardware level, e.g. by shielding blocks with ground lines).

6.2.3.1 Security of MAC_1

The following theorem establishes the LR-security of our MAC_1 construction.

Theorem 6.1. *Let $\mathsf{F} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be an (s, t, ϵ_{prf}) PRF having $(s_{\mathcal{S}}, t_{\mathcal{S}}, s, t, \epsilon_{2\text{-}sim})$ 2-simulatable leakages. Then, the instantiation of MAC_1 with F is an (q, s', t', ϵ')-strongly-unforgeable LR-MAC on messages of length l with n-bit blocks, where:*

$$\epsilon' \leq \epsilon_{prf} + (q+1)l(\epsilon_{prf} + \epsilon_{2\text{-}sim}) + negl(n),$$

with $s' \approx s - q \cdot l \cdot s_{\mathcal{S}}$ and $t' \approx t - \tilde{t}$, where $\tilde{t}$ is the time required by the challenger to simulate the experiment $\mathsf{Forge}^{\mathsf{euf-cma}}_{\mathcal{A}^{\mathsf{L}},\mathsf{MAC}}(n)$ for the construction MAC_1, which

essentially consists of $(q+1)(l+1)$ *evaluations of* F *and* $q \cdot l$ *calls to the simulator* $\mathcal{S}^{\mathsf{L}}$. *Here,* $negl(n)$ *refers to a negligible function of* n *assuming that* q *and* l *are polynomially bounded in* n.

Proof. The proof of strongly-unforgeable LR-MAC security for our MAC_1 construction proceeds by a sequence of hybrid games, which essentially follows the strategy introduced in [SPY13, Theorem 1].

Consider the following hybrid games:

Hybrid H^{+}. This is the original security game executed as defined in the experiment $\mathsf{Forge}^{\mathsf{euf-cma}}_{\mathcal{A}^{\mathsf{L}},\mathsf{MAC}}(n)$ (Definition 6.1, Remark 6.1). In particular, the q session keys $k_1^{(i)}$ $(i = 1, 2, \ldots, q)$ are the output of F on random IVs. Let the queried messages (each having l blocks) be denoted as $m_i = \left\langle m_1^{(i)}, m_2^{(i)}, \ldots, m_l^{(i)} \right\rangle$ $(i = 1, 2, \ldots, q)$. The advantage of a q-query adversary $\mathcal{A}^{\mathsf{L}}$ in $\mathsf{Forge}^{\mathsf{euf-cma}}_{\mathcal{A}^{\mathsf{L}},\mathsf{MAC}}(n)$ is ϵ'.

Hybrid H^{++}. This is the same as hybrid H^{+} except that the q IVs randomly chosen are distinct. Let ϵ^{++} denote the advantage of $\mathcal{A}^{\mathsf{L}}$ in this hybrid. Using the birthday bound, we have that

$$\left|\epsilon' - \epsilon^{++}\right| \leq \frac{q^2}{2^{n+1}}. \tag{6.1}$$

Hybrid H^{*}. This is the same as hybrid H^{++} except that the q session keys $k_1^{(i)} \xleftarrow{\$} \{0,1\}^n$ are chosen uniform randomly and independently. Let ϵ^{*} denote the advantage of $\mathcal{A}^{\mathsf{L}}$ in this hybrid. It is easy to see that

$$\left|\epsilon^{++} - \epsilon^{*}\right| \leq \epsilon_{prf}. \tag{6.2}$$

This is because using the adversary $\mathcal{A}^{\mathsf{L}}$ of MAC_1, we can construct a (s, t', ϵ_{prf}) against F.

Hybrids $H^{*}_{i,j}$. Next, we successively transform the hybrid H^{*} into hybrids $H^{*}_{i,j}$ $(1 \leq i \leq q+1, 0 \leq j \leq l)$ by transforming the normal execution of the block cipher F into an ideal execution one message block at a time. More precisely, the hybrids H^{*} and $H^{*}_{1,0}$ are identical. In hybrid $H^{*}_{1,1}$, the output of $\mathsf{F}(k_1^{(1)}, m_1^{(1)})$ while processing the first message block during the first tag-generation query is a uniform random element in $\{0,1\}^n$ and leakages are simulated (cf. Lemma 6.1). The processing of the remaining message blocks in the first as well as the later queries is carried out "normally." By "normal" we mean that the actual values of F are used for all except the first block of the first message and all the session keys, unless for consistency we are forced to use the random sampled value in case the inputs $(k_1^{(1)}, m_1^{(1)})$ to F appear again elsewhere later.

More generally, in the hybrid $H^*_{i,j}$, all the evaluations of F are treated as ideal until (and including) the j^{th} message block of the i^{th} tag query. All the remaining evaluations of F are normal upto the consistency requirement mentioned above. The hybrids $H^*_{i,l}$ and $H^*_{i+1,0}$ are identical for $1 \leq i \leq q$, and the verification of the plausible forgery output by $\mathcal{A}^{\mathsf{L}}$ is considered as $q+1$-st tag query except that the leakages and the tag are not output. This means that the hybrids $H^*_{q+1,j}$ correspond to idealizing the execution of F during the verification stage. Note that the goal of the adversary $\mathcal{A}^{\mathsf{L}}$ is to break the strong-unforgeability of MAC_1. This means that either it outputs a forgery on a distinct IV for a possibly previously queried message, or it outputs a forgery on a previously queried IV but on a distinct message. We assume without loss of generality that the forgery output by $\mathcal{A}^{\mathsf{L}}$ satisfies either of the above two conditions and it makes exactly q tag request queries. Hence the hybrids $H^*_{q+1,j}$ will be present. Note that this last sequence of hybrids may be avoided if we try to follow the standard approach of first showing that the construction is a PRF and hence it is a MAC. But it turns out this way we need more hybrid games than the current approach.

Next, we show that in the successive hybrids $H^*_{i,j}$ and $H^*_{i,j+1}$ ($1 \leq i \leq q+1, 0 \leq j \leq l-1$), the views of $\mathcal{A}^{\mathsf{L}}$ are computationally identical upto the 2-simulatable leakage assumption. Let $\epsilon_{i,j}$ and $\epsilon_{i,j+1}$ denote the advantages of $\mathcal{A}^{\mathsf{L}}$ in the hybrids $H^*_{i,j}$ and $H^*_{i,j+1}$, respectively.

Lemma 6.2. $|\epsilon_{i,j} - \epsilon_{i,j+1}| \leq \epsilon_{prf} + \epsilon_{2\text{-}sim} + \frac{(q+1)^2(l+1)^2}{2^n}$.

Proof. Using $\mathcal{A}^{\mathsf{L}}$ as a subroutine, we construct a (s', t')-bounded distinguisher $\mathcal{D}^{\mathsf{L}}$ for the distributions of Lemma 6.1 that has advantage as indicated on the R.H.S. of Lemma 6.2. $\mathcal{D}^{\mathsf{L}}$ simulates hybrid $H^*_{i,j}$ or hybrid $H^*_{i,j+1}$ depending on whether its input distribution is actual or ideal, respectively.

$\mathcal{D}^{\mathsf{L}}$ chooses a uniform random shared secret key k, random session keys $k_{i,1}$ ($1 \leq i \leq q$), and possibly a random session key $k_{q+1,1}$ for verification if the target forgery is on a different IV. Whenever $\mathcal{D}^{\mathsf{L}}$ samples a random output value, say γ, on "key-message" input (α, β) it records the input/output pair $((\alpha, \beta), \gamma)$ in a table $\mathcal{T}$, in addition to the simulated leakage $\mathcal{S}^{\mathsf{L}}(\tilde{k}^*, \beta, \gamma)$ ($\tilde{k}^* \xleftarrow{\$} \{0,1\}^n$). This table is used to consistently return the same (random) output on the same input pair. Also, all the block cipher evaluations while the processing the tag requests return random outputs and simulated leakages until (including) the j^{th} message block of the i^{th} tag request. Recall the notation that the j'^{th} block of the i'^{th} message is denoted by $m_{i',j'}$, and the corresponding key and the output for the evaluation of F is denoted by $k_{i',j'}$ and $k_{i',j'+1}$, respectively. A '*' in the superscript of a parameter, for example, as in $k^*_{i',j'}$, explicitly denotes that the parameter was chosen uniform randomly and independently (and was not computed normally).

At this point, $\mathcal{D}^{\mathsf{L}}$ first receives its input d_5 upon querying its challenger with $\mathsf{Gen}(\tilde{k}^*_{i,j}, m_{i,j})$ with $\tilde{k}^*_{i,j} \xleftarrow{\$} \{0,1\}^n$ (cf. Remark 6.3). It then uses d_5 instead of

the simulated leakage $\mathcal{S}^{\mathsf{L}}(\tilde{k}^*_{i,j}, m_{i,j}, k^*_{i,j})$. Note that the output of this round is implicitly set to k' (if $j = 0$, then the session key $k^*_{i,1}$ is implicitly set to k'). It then builds the view for the $(i, j+1)^{\text{th}}$ execution of F by querying its challenger for (d_1, d_3) by querying $\mathsf{Enc}(m_{i,j+1})$ ($p_0 := m_{i,j+1}$). $\mathcal{D}^{\mathsf{L}}$ then provides $\mathcal{A}^{\mathsf{L}}$ with (d_1, d_3). The remaining steps are executed normally by evaluating F with the (known) inputs. In case any inconsistency arises when F is evaluated with the inputs present in the table $\mathcal{T}$, then the corresponding output value recorded in the table is used. Such a situation does arise when the adversary outputs a possible forgery on a message m' w.r.t. the IV IV_i, such that $m' \neq m_i$ share a common prefix. Also, note that to evaluate F w.r.t. the (implicit) key k', $\mathcal{D}^{\mathsf{L}}$ has to use its input distribution to determine the output. This means that if there are more than two queries to F w.r.t. the (implicit) key k', then $\mathcal{D}^{\mathsf{L}}$ will abort the simulation. Denote this abort event by Abort.

In order for this simulation by $\mathcal{D}^{\mathsf{L}}$ to $\mathcal{A}^{\mathsf{L}}$ to be consistent with the working of MAC_1, two events must *not* occur.

- All the random values sampled (and listed in table $\mathcal{T}$) must be distinct.
- The abort event Abort mentioned above must not occur.

Let us denote both the events by $\mathsf{Collision}$. The reason for the first of the above conditions is that if the intermediate values repeat, then a possible pattern could arise in the successive intermediate values and there by implicitly setting the output of the $(i, j)^{\text{th}}$ step to k' may lead to inconsistency. Next, to ensure that the Abort event does not arise, and hence 2-simulatable assumption suffices, we need to make sure that k' is never used as key more than once later. This means that F is not queried with the input $(k^*_{i,j}, m_{i,j})$ more than once later. Note that if the first condition above does not occur, then the outputs of the later steps are function of parameters independent of the value $k^*_{i,j}$, except possibly once during the verification stage. It is easy to see that

$$\Pr[\mathsf{Collision}] \leq \frac{(q+1)^2(l+1)^2}{2^n}. \tag{6.3}$$

Hence the lemma follows from Lemma 6.1 and (6.3).

□

Note that there are at most $(q+1)l$ *distinct* hybrids $H^*_{i,j}$ ($1 \leq i \leq q+1, 0 \leq j \leq l$) since the hybrids $H^*_{i,l}$ and $H^*_{i+1,0}$ are identical for $1 \leq i \leq q$. Also note that the hybrids H^* and $H^*_{1,0}$ are identical as well. Hence we obtain

$$|\epsilon^* - \epsilon_{q+1,l}| \leq (q+1)l(\epsilon_{prf} + \epsilon_{2-sim}) + \frac{(q+1)^3 l(l+1)^2}{2^n}. \tag{6.4}$$

Lemma 6.3. $\epsilon_{q+1,l} \leq \frac{1}{2^n - ((q+1)(l+1))^2} + \frac{(q+1)^2(l+1)^2}{2^{n+1}}$

Proof. In this hybrid, all the evaluations of F are ideal, that means that distinct "key-message" input pairs produce random output values, and all the leakages are simulated. To break the strong-unforgeability property of MAC_1, $\mathcal{A}^{\mathsf{L}}$ must either output a forgery on a distinct IV for a possibly previously queried message, or it outputs a forgery on a previously queried IV but on a distinct message. This implies that during the verification step, conditioned on the event of no collision of randomly chosen output values, there will be distinct "key-message" pairs with which F will be queried and, as a consequence, a random output will be produced. Further, no collision would imply that the same would happen with the later message blocks during the verification step, including the final block. Note that the probability of collision is at most $\frac{(q+1)^2(l+1)^2}{2^{n+1}}$. Again, conditioned on the event of no collision, the probability that the tag τ output by $\mathcal{A}^{\mathsf{L}}$ is a valid forgery for the message m is at most $\frac{1}{2^n-((q+1)(l+1))^2}$. Hence the lemma follows.

□

From (6.1), (6.2), (6.4) and Lemma 6.3, the Theorem 6.1 follows.

□

Remark 6.5 A glance at Figure 6.2 might suggest that only the 1-simulatability leakage assumption would suffice for the security of the MAC_1 construction, but this does not seem to be the case. Indeed, for most parts of the security reduction, the 1-simulatability leakage assumption is enough. But because we allow the adversary to possibly output a forgery on a previously used IV, we need the second output pair of the leakage simulator to enable us to verify the attempted forgery by the adversary (on this particular IV). Note that this issue only relates to the reduction and has nothing to do with the construction itself (for which we exclude the leakage during verification).

6.2.4 Simplification: the Hash then MAC Paradigm

To conclude this section, we note that in view of how the message in Figure 6.2 is processed, an alternative (and in fact even simpler) solution to build a leakage-resilient MAC is to rely on the hash then MAC paradigm. Such a proposal is intuitively depicted in Figure 6.3, where we can see that the leakage-resilience of the scheme now really boils down to the execution of the leak-free block cipher on a random IV, which comes at the cost of an additional building block (namely a collision-resistant hash function). This essentially results from the fact that the hash function is only executed on public inputs. Interestingly, this construction also suggests that a standard solution like HMAC could be slightly tweaked in order to become leakage-resilient (which is in contrast with the previously mentioned CBC-MAC).

More precisely, our construction $\mathsf{MAC}_2 = (\mathsf{KeyGen}_2, \mathsf{Mac}_2, \mathsf{Vrfy}_2)$, can be viewed as a special instantiation of MAC_1 where the messages of arbitrary length are first hashed to a single n-bit block using the hash function H :

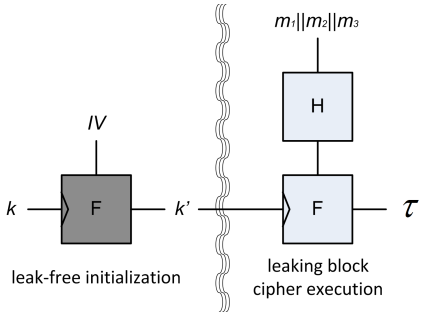

Figure 6.3: Hash then MAC.

$\{0,1\}^* \to \{0,1\}^n$. Then a tag is generated for this hashed block using MAC_1, which makes its analysis straightforward.

Description of MAC_2:

- $\mathsf{KeyGen_2}(1^n)$: Choose a hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^n$, and a shared symmetric-key $k \xleftarrow{\$} \{0,1\}^n$.
- $\mathsf{Mac_2}\,(m,k)$: Choose $IV \xleftarrow{\$} \{0,1\}^n$. Compute $k' = \mathsf{F}_k(IV)$, $h = \mathsf{H}\,(m)$, and $r = \mathsf{F}_{k'}\,(h)$. **Return** $\tau = (IV, r)$.
- $\mathsf{Vrfy_2}\,(m,\tau,k)$: Parse $\tau = (IV, tg)$. Compute $\tau' \leftarrow \mathsf{Mac}_2\,(m,k,IV)$.
  - If $\tau' \stackrel{?}{=} \tau$, then **return** 1 (correct), else **return** 0 (incorrect).

Our proof requires the definition of a collision-resistant hash function.

Definition 6.4. [Collision Resistance]. *A hash function* $\mathsf{H} : \{0,1\}^* \to \{0,1\}^n$ *is said to be* (t, ϵ_{cr}) *collision-resistant if for any adversary* $\mathcal{A}$ *running for time at most* t*, its advantage in outputting* $m, m' \in \{0,1\}^*$ *such that* $m \neq m'$ *and* $\mathsf{H}(m) = \mathsf{H}(m')$*, is at most* ϵ_{cr}*.*

Based on this definition, the following theorem establishes the LR-security of our MAC_2 construction.

Theorem 6.2. *Let* $\mathsf{F} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ *be an* (s, t, ϵ_{prf}) *PRF having* $(s_{\mathcal{S}}, t_{\mathcal{S}}, s, t, \epsilon_{2\text{-}sim})$ *2-simulatable leakages, and let* $\mathsf{H} : \{0,1\}^* \to \{0,1\}^n$ *be a* $(t, \epsilon_{\mathsf{cr}})$ *collision-resistant hash function. Then, the instantiation of* MAC_2 *with* F *and* H *is an* (q, s', t', ϵ')*-strongly-unforgeable LR-MAC on messages of* arbitrary length*, where:*

$$\epsilon' \leq \epsilon_{cr} + \epsilon_{prf} + (q+1)(\epsilon_{prf} + \epsilon_{2\text{-}sim}) + negl(n),$$

with $s' \approx s - q \cdot s_{\mathcal{S}}$ *and* $t' \approx t - \tilde{t}$*, where* $\tilde{t}$ *is the time required by the challenger to simulate the experiment* $\mathsf{Forge}^{\mathsf{euf-cma}}_{\mathcal{A}^{\mathsf{L}},\mathsf{MAC}}(n)$ *for the construction* MAC_2*, which essentially consists of* $2(q+1)$ *evaluations of* F *and* q *calls to the simulator* $\mathcal{S}^{\mathsf{L}}$*. Here,* $negl(n)$ *refers to a negligible function of* n *assuming that* q *and* l *are polynomially bounded in* n*.*

Proof sketch. We just observe that if the adversary is unable to break the collision resistance of H, then it has to output a valid forgery on a new hash output (corresponding to some message) for a previously queried IV, or on an old hash output but for a new IV. By treating the n-bit hash outputs as the message space of MAC_1, we obtain the above bound on the advantage of $\mathcal{A}$ from Theorem 6.1 (with $l = 1$). Note that the adversary's advantage in breaking the collision-resistance of H is ϵ_{cr}. □

6.3 Leakage-Resilient Encryption

6.3.1 Security Definition

We now turn to the construction of a leakage-resilient symmetric encryption scheme ENC with key generation algorithm Gen, encryption algorithm Enc and decryption algorithm Dec. The Enc algorithm proceeds on messages made of a variable number of blocks, i.e. messages from a set $(\{0,1\}^n)^*$ where n is the block size.

For this scheme, we define an $\mathsf{PrivK}^{\mathsf{lmcpa,b}}_{\mathcal{A}^{\mathsf{L}},\mathsf{ENC}}$ game, analogue to the traditional IND-CPA security game, but in a physical setting where all encryption operations, including the test query, return to the adversary a leakage together with a ciphertext. This game is described in Table 6.1.

$\mathsf{PrivK}^{\mathsf{lmcpa,b}}_{\mathcal{A}^{\mathsf{L}},\mathsf{ENC}}$ is the output of the following experiment:

1. A key k is generated by running Gen.
2. $\mathcal{A}^{\mathsf{L}}$gets access to a leaking encryption oracle that, on messages of arbitrary block length, returns an encryption of these messages together with the leakage resulting from the encryption process.
3. $\mathcal{A}^{\mathsf{L}}$ submits two messages m_0 and m_1 of identical block length
4. A ciphertext $c \leftarrow \mathsf{Enc}_k(m_b)$ is computed, resulting in a leakage l. Both c and l are given to $\mathcal{A}^{\mathsf{L}}$.
5. $\mathcal{A}^{\mathsf{L}}$ can keep accessing the leaking encryption oracle.
6. $\mathcal{A}^{\mathsf{L}}$ outputs a bit b'.

Table 6.1: Multiple block leakage-resilient CPA game for symmetric encryption.

Naturally, we will be interested in the difference $|\mathsf{PrivK}^{\mathsf{lmcpa,0}}_{\mathcal{A}^{\mathsf{L}},\mathsf{ENC}} - \mathsf{PrivK}^{\mathsf{lmcpa,1}}_{\mathcal{A}^{\mathsf{L}},\mathsf{ENC}}|$, which we would like to be minimal. However, and as discussed in the introduction, since we consider leakages even during the test query, we cannot expect this difference to be negligible. For that reason, we rather focus on establishing bounds that are derived from the security of a considerably simpler encryption scheme, that encrypts only one message made of a single block per key. That is, we want to show that any security security guarantee that can

be ensured for this simple (one-time, single-block) encryption scheme (next denoted by $\mathsf{ENC1}$) extends to our full construction ENC. This is eventually what we achieve in Theorems 6.3 and 6.4, which relate the CPA security of the multi-block ENC scheme to the eavesdropper security of the single-block $\mathsf{ENC1}$ scheme. We believe that such a result helps the task of secure implementation and security evaluation in two ways:

1. The eavesdropper security game gives a unique leakage for a single-block message to the adversary, which is a most limited input to run a side-channel attack (such attacks usually rely on a few hundred leakages). This means that a cheap and relatively weakly protected implementation could be used even in a setting where long messages need to be encrypted [BGS15].
2. Security evaluations can also focus on the (comparatively) simpler task of assessing the security of a single round of encryption, without needing to care that the combination of leakages from the encryption of multiple blocks of message could become a problem. (For instance, leaking one bit of a key per encryption block is probably not a problem when encrypting a single block, but could become a problem if the encryption of each block leaks a new bit. Our proof guarantees that there is no such risk.)

The rest of this section is organized as follows. We start in Section 6.3.2 by defining our leakage-resilient encryption scheme ENC and its leakage model, in the spirit of our MAC treatment. Next, in Section 6.3.3, we define our one-time and one-block encryption scheme $\mathsf{ENC1}$, together with its leakage model.

Based on these definitions, we build our security analysis as follows. In Section 6.3.4, we define an idealized version $\mathsf{ENC1}^{\mathsf{I}}$ of $\mathsf{ENC1}$ that has perfectly random outputs and simulated leakages for the PRFs. We also define a one-time (but multiple block) version of the ENC scheme, which we call ENCl (the l referring to the l blocks), as well as an ideal version $\mathsf{ENCl}^{\mathsf{I}}$ of it. We conclude this section by bounding the probability that an adversary distinguishes between "real" and ideal version of the schemes. In Section 6.3.5, we push our analysis one step further, bounding the probability that an adversary breaks eavesdropper security for the $\mathsf{ENCl}^{\mathsf{I}}$ scheme as a function of the probability of breaking that same property on the $\mathsf{ENC1}^{\mathsf{I}}$ scheme. The result from Section 6.3.4 can then be used to move back to the "real" encryption schemes. Eventually, in Section 6.3.6, we conclude by relating the CPA security of the ENC scheme to the eavesdropper security of the ENCl scheme.

6.3.2 Encryption with a Leakage-Resilient Stream Cipher

The ENC scheme. Our starting point is the leakage-resilient stream cipher from [SPY13], which we transform into an encryption scheme by XORing its output with the message to be encrypted. CPA security is obtained by adding

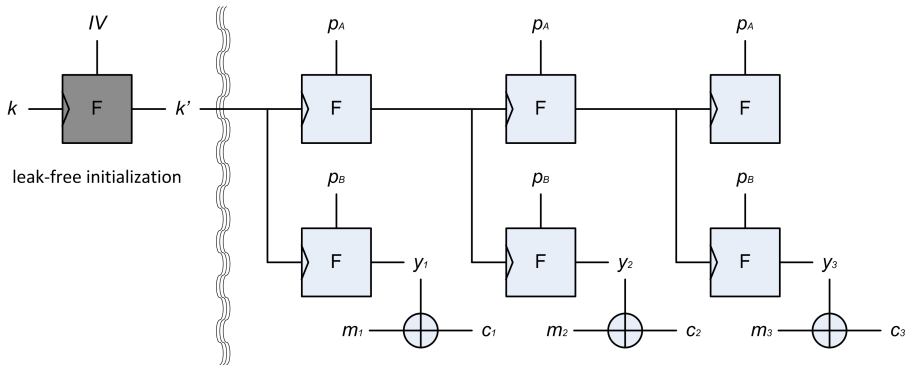

Figure 6.4: Encryption with a leakage-resilient stream cipher.

an initialization round, which generates the stream-cipher seed by applying a leak-free PRF, keyed with the encryption key, to an initialization vector IV, as represented in Figure 6.4. As for the previous MAC constructions, we require the initialization step be leak-free in order to make sure that, despite the fact that it will be executed many times with the same key, that key will remain safe. And here as well, this use is minimal (a single execution per message) and independent of the message (so that the fresh key k' can possibly be pre-computed.) The ENC encryption scheme is defined more formally in Table 6.2.

- Gen picks a random key $k \leftarrow \{0,1\}^n$.
- Enc picks a random $IV \leftarrow \{0,1\}^n$, then computes $k' := k_0 = \mathsf{F}^*_k(IV)$ using the leak-free PRF. The encryption of the l-block message $m = m_1, \ldots, m_l$ is then computed as $IV, c_1, \ldots, c_l$, where $c_i = y_i \oplus m_i$, $y_i = \mathsf{F}_{k_{i-1}}(p_B)$ and $k_i = \mathsf{F}_{k_{i-1}}(p_A)$ ($p_A \neq p_B$ are public constants).
- Dec proceeds in the natural way.

Table 6.2: The ENC encryption scheme

Leakage model and Assumptions. We capture the leakages of an implementation of this encryption scheme through two leakage functions: $\mathsf{L}_\mathsf{F}(p,k)$ that defines the leakage of each PRF running on plaintext p with key k, and $\mathsf{L}_\oplus(m,y)$ that defines the leakage of computing the XOR of m and y. (When the adversary $\mathcal{A}^\mathsf{L}$ has the single L superscript, we mean that it can query both these leakage functions.) So, the encryption of each message block m_i causes the following leakages: $\mathsf{L}_\mathsf{F}(p_A, k_{i-1})$, $\mathsf{L}_\mathsf{F}(p_B, k_{i-1})$ and $\mathsf{L}_\oplus(m_i, y_i)$. Here and later, we precede an algorithm with the L letter (e.g. $\mathsf{LEnc}_k(m)$) to refer to both the output of an encryption and the resulting leakage.

As in the previous section about MACs, we require the leakages of the PRF

to be 2-simulatable, but no more. As a consequence, leakage functions do not need to be efficient, and can possibly leak several bits of information on their inputs. In particular, they can provide several bits of information on y_i and m_i, which makes traditional security notions based on indistinguishability impossible to achieve. We believe that, without additional leak-free component, this is just unavoidable: at some point, messages need to be used during the encryption process, and this use must be expected to leak information that is sufficient to win any indistinguishability game.

6.3.3 A Single-block One-time Encryption Scheme

Description of $\mathsf{ENC1}$:	Description of $\mathsf{ENC1}^{\mathsf{l}}$:
– Gen picks $k_0 \leftarrow \{0,1\}^n$. – $\mathsf{Enc1}_{k_0}(m)$ returns (k_1, c_1), where $c_1 = y_1 \oplus m$, $y_1 = F_{k_0}(p_B)$ and $k_1 = F_{k_0}(p_A)$. – Dec proceeds in the natural way. The leakage resulting from $\mathsf{Enc1}_{k_0}(m)$ is defined as $\mathsf{L}_{\mathsf{Enc1}}(k_0, m) := (\mathsf{L}_{\mathsf{F}}(p_A, k_0), \mathsf{L}_{\mathsf{F}}(p_B, k_0), \mathsf{L}_{\oplus}(m, y_1), \mathcal{S}^{\mathsf{L}}(k^-, p_A, k_0), k^-)$ with $k^- \leftarrow \{0,1\}^n$.	– $\mathsf{Enc1}^{\mathsf{l}}{}_{k_0}(m)$ returns (k_1, c_1), where $c_1 = y_1 \oplus m$, $y_1 \leftarrow \{0,1\}^n$ and $k_1 \leftarrow \{0,1\}^n$. The leakage resulting from $\mathsf{Enc1}^{\mathsf{l}}{}_{k_0}(m)$ is defined as $\mathsf{L}_{\mathsf{ENC1}^{\mathsf{l}}}(k_0, k_1, y_1, m) := (\mathcal{S}^{\mathsf{L}}(k_0, p_A, k_1), \mathcal{S}^{\mathsf{L}}(k_0, p_B, y_1), \mathsf{L}_{\oplus}(m, y_1), \mathcal{S}^{\mathsf{L}}(k^-, p_A, k_0), k^-)$ with $k^- \leftarrow \{0,1\}^n$.

Table 6.3: The $\mathsf{ENC1}$ and $\mathsf{ENC1}^{\mathsf{l}}$ schemes.

We define, in the left part of Table 6.3, the $\mathsf{ENC1}$ single-block one-time encryption scheme, from the security of which we will infer the $\mathsf{PrivK}^{\mathsf{lmcpa}}_{\mathcal{A}^{\mathsf{L}},\mathsf{ENC}}$ security of our ENC multi-block scheme. The $\mathsf{ENC1}$ scheme, while being quite similar to a single block version of ENC, also bears important differences with ENC:

1. It has no leak-free initialization process. This is not necessary for a one-time version of the scheme.
2. Its ciphertext contains k_1. While being harmless from a black-box point of view, including k_1 in the ciphertext will show to be useful in bounding the amount of information that leakages can transfer between rounds. We will provide constructive evidence that this is necessary after the proof of Lemma 6.5.
3. Its leakages contain $\mathcal{S}^{\mathsf{L}}(k^-, p_A, k_0), k^-)$ with a random k^-. This leakage has the same purpose as k_1 in the ciphertext, and will be used to bound the information that can leak, in the multi-block setting, from the encryption of previous blocks.

6.3.4 One-time Ideal Versions of Our Encryption Schemes

We now idealize the $\mathsf{Enc1}$ encryption process by replacing the use of F for computing k_1 and y_1 by the selection of random values. Furthermore, we adapt the corresponding leakages using $\mathcal{S}^{\mathsf{L}}$. The resulting algorithms are defined in the right part of Table 6.3. The following lemma expresses that leaking encryptions produced with these two schemes are hard to distinguish, by relying on the 2-simulatability assumption and the properties of the PRF.

Lemma 6.4. Ideal single block encryption. *Let* $\mathsf{F} : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ *be a* (s, t, ϵ_{prf})*-PRF, whose implementation has a leakage function* $\mathsf{L_F}$ *having* $(s_{\mathcal{S}}, t_{\mathcal{S}}, s, t, \epsilon_{2\text{-}sim})$ 2*-simulatable leakages, and let* $\mathcal{S}^{\mathsf{L}}$ *be an appropriate* $(s_{\mathcal{S}}, t_{\mathcal{S}})$*-bounded leakage simulator. Then, for every* $m, p_A, p_B, p \in \{0,1\}^n$ $(p_A \neq p_B)$ *and every* $(s - s_r,\ t - t_r)$*-bounded distinguisher* $\mathcal{D}^{\mathsf{L}}$*, the following holds:*

$$\Big|\Pr\left[\mathcal{D}^{\mathsf{L}}\left(m, \mathsf{LEnc1}(k_0, m)\right) = 1\right] - \Pr\left[\mathcal{D}^{\mathsf{L}}\left(m, \mathsf{LEnc1}^{\mathsf{I}}(k_0, m)\right) = 1\right]\Big| \leq \epsilon_{prf} + \epsilon_{2\text{-}sim},$$

with $s_r := 3s_{\mathcal{S}} + 1$ *and* $t_r = \max(t_{prf}, t_{sim})$ *with* t_{prf} *being equal to* $3t_{\mathcal{S}}$ *augmented with the time needed to make* 2 *oracle queries to the PRF challenger and select a uniformly random key in* $\{0,1\}^n$*, and* t_{sim} *being the time needed to relay the content of two* Enc *and one* Gen *queries from and to a q-sim challenger.*

Proof. We follow the approach used for proving Lemma 6.1: first replace the leakages of $\mathsf{LEnc1}$ with simulated leakages, relying on the simulatability assumption, then replace the outputs of the PRF of $\mathsf{LEnc1}$ with random values, relying on the assumption that F is a PRF. □

We now transpose the definitions of $\mathsf{ENC1}$ and $\mathsf{ENC1}^{\mathsf{I}}$ to the multi-block setting, but still focusing on the one-time encryption case. The resulting schemes, ENCI and $\mathsf{ENCI}^{\mathsf{I}}$ are described in Table 6.4. These ideal versions are closer to the ENC scheme definition: while we still ignore the leak-free initialization process, the ciphertexts do not contain the extra key block any more, and the leakages follow the natural definition.

Just as before, we express that leaking encryptions produced with these two schemes are hard to distinguish.

Lemma 6.5. Ideal multiple block encryption. *Let* F *and* $\mathcal{S}^{\mathsf{L}}$ *be defined as in Lemma 6.4. Then, for every* l*-block message* m*, every* p_A, p_B $(p_A \neq p_B)$ *and every* $(s - s_r, t - t_r)$*-bounded distinguisher* $\mathcal{D}^{\mathsf{L}}$*, the following holds:*

$$\Big|\Pr\left[\mathcal{D}^{\mathsf{L}}\left(m, \mathsf{LEncl}(k_0, m))\right) = 1\right] - \Pr\left[\mathcal{D}^{\mathsf{L}}\left(m, \mathsf{LEncl}^{\mathsf{I}}(k_0, m)\right) = 1\right]\Big| \leq l(\epsilon_{prf} + \epsilon_{2\text{-}sim}) + negl(n).$$

Description of ENCI:	Description of $\mathsf{ENCI}^{\mathsf{I}}$:
– Gen picks $k_0 \leftarrow \{0,1\}^n$. – $\mathsf{Encl}_{k_0}(m_1,\ldots,m_l)$ returns $c_1,\ldots,c_l$, where $c_i = y_i \oplus m_i$, $y_i = \mathsf{F}_{k_{i-1}}(p_B)$ and $k_i = \mathsf{F}_{k_{i-1}}(p_A)$. – Dec proceeds in the natural way. The leakage $\mathsf{L}_{\mathsf{Encl}}(k_0, m)$ resulting from computing $\mathsf{Encl}_{k_0}(m)$ is defined by the sequence of $(\mathsf{L}_{\mathsf{F}}(p_A, k_{i-1}), \mathsf{L}_{\mathsf{F}}(p_B, k_{i-1}), \mathsf{L}_{\oplus}(m_i, y_i))$ for $i \in \{1,\ldots,l\}$.	– $\mathsf{Encl}^{\mathsf{I}}{}_{k_0}(m_1,\ldots,m_l)$ returns $(c_1,\ldots,c_l)$, where $c_i = y_i \oplus m_i$, $y_1,\ldots,y_l \leftarrow \{0,1\}^n$ and $k_1,\ldots,k_l \leftarrow \{0,1\}^n$. The leakage $\mathsf{L}_{\mathsf{Encl}^{\mathsf{I}}}(k, y, m)$ resulting from computing $\mathsf{Encl}^{\mathsf{I}}{}_{k_0}(m)$ with the random vectors k and y is defined by the sequence of $(\mathcal{S}^{\mathsf{L}}(k_{i-1}, p_A, k_i), \mathcal{S}^{\mathsf{L}}(k_{i-1}, p_B, y_i), \mathsf{L}_{\oplus}(m_i, y_i))$ for $i \in \{1,\ldots,l\}$.

Table 6.4: The ENCI and $\mathsf{ENCI}^{\mathsf{I}}$ schemes.

Here, $s_r = l(2s_{\mathcal{S}} + 3)$ and t_r is equal to $2lt_{\mathcal{S}}$ augmented with the time needed to pick $2l$ random values in $\{0,1\}^n$, evaluate F $2l$ times and compute l $\oplus$ operations. And, $negl(n)$ refers to a negligible function of n assuming that l is polynomially bounded in n.

Proof. We define the hybrid distributions $H_0,\ldots,H_l$ where $H_i(m)$ is computed as the concatenation of $\mathsf{LEncl}^{\mathsf{I}}{}_{k_0}(m_{[1,i]})$ and $\mathsf{LEncl}_{k_i}(m_{[i+1,l]})$ with k_0 chosen uniformly at random and k_i resulting from the evaluation of $\mathsf{Encl}^{\mathsf{I}}$. It is clear that H_0 is distributed just as the inputs of $\mathcal{D}^{\mathsf{L}}$ in the first probability from the lemma's statement, while H_l is distributed as the inputs of $\mathcal{D}^{\mathsf{L}}$ in the second probability.

We now show that the probability with which $\mathcal{D}^{\mathsf{L}}$ can distinguish H_{i-1} from H_i is bounded by $\epsilon_{prf} + \epsilon_{2\text{-}sim}$, which will in turn imply the expected result. To this purpose, we build, from $\mathcal{D}^{\mathsf{L}}$, a (s,t)-bounded distinguisher $\mathcal{D}^{\mathsf{L}\prime}$ between the two distributions d_1 and d_2 that are the input of the distinguisher of Lemma 6.4. $\mathcal{D}^{\mathsf{L}\prime}$ receives its inputs m_i, c_i, k_i, and leakages l_A, l_B, $l_{\oplus}$, l_s, k_{i-2} sampled from d_1 or d_2. It then:

1. Samples the encryption of the $i-1$ first blocks of m from $\mathsf{LEncl}^{\mathsf{I}}$ by choosing random keys k_j, except that it uses k_{i-2} as key for the round $i-1$ and l_s as leakage for computing k_{i-1};
2. Extends it with c_i and the leakages l_A, l_B and $l_{\oplus}$ for the i-th round;
3. Extends it with $\mathsf{LEncl}_{k_i}(m_{[i+1,l]})$ for the last $l-i-1$ rounds.

It can be easily verified that, if the inputs of $\mathcal{D}^{\mathsf{L}\prime}$ are sampled from d_1 (resp d_2), then $\mathcal{D}^{\mathsf{L}\prime}$ produced something sampled according to H_{i-1} (resp. H_i), provided that distinct round keys are chosen and distinct round keys are later on (deterministically) computed (in or) after round i. This restriction arises out of the need to sample random outputs for the first $i-1$ rounds while

maintaining the input-output consistency (similar event, Collision, appears in the proof of Lemma 6.2). The probability of this event is upper bounded by $l^3/2^n$.

If fed to $\mathcal{D}^{\mathsf{L}}$, the result will enable $\mathcal{D}^{\mathsf{L}'}$ to distinguish H_{i-1} from H_i with the same probability $\mathcal{D}^{\mathsf{L}'}$ distinguishes d_1 from d_2, unless the above mentioned collision event has occurred. Furthermore, by inspection, we can verify that $\mathcal{D}^{\mathsf{L}'}$ is (s,t)-bounded. Applying Lemma 6.4, this probability is then bounded by $\epsilon_{prf} + \epsilon_{2\text{-}sim} + l^3/2^n$, as desired.

□

Further Remarks on the Definition of Enc1. The proof above heavily relies on the use of the extra leakages provided when running Enc1, for the linking of the hybrids. This is however not just an artifact that we use to simplify our proof. Consider for instance a situation in which Enc1 would not leak k_1 and a simple leakage function $\mathsf{L}_{\mathsf{F}}(p,k)$ would leak just the first bit of $k \oplus \mathsf{F}_k(p)$. In such a setting, if k_1 is not leaked, the leakage does not provide any useful information on the encrypted message (we just loose one bit of security for the key). So, if we encrypt the messages m_1 and m_2 with Enc1 using two independent keys, the leakages do not provide us with any useful information. However, if we encrypt the message (m_1, m_2) using Encl, we will obtain c_1, c_2 and leakages containing the first bit of $k_0 \oplus y_1$, $k_0 \oplus k_1$ and $k_1 \oplus y_2$, from which we can derive the first bit of $y_1 \oplus y_2$, and eventually the first bit of $m_1 \oplus m_2$, which was not available before. This observation is a constructive evidence that encrypting two message blocks with Encl can be far more damaging on the privacy than encrypting these blocks independently with a version of Enc1 from which k_1 would not be leaked. The leakage of k_1 in Enc1 prevents the shortcoming we just described, as k_1 would provoke the leakage of the first bit of each message block.

6.3.5 From 1-block to l-block Eavesdropper Security

The above section demonstrated how one-time versions of our encryption scheme can be idealized with controlled security loss, in the case of single and multiple block encryption. We now use these idealized encryption processes to evaluate the (eavesdropper) security of an l-block encryption with $\mathsf{Encl}^{\mathsf{l}}$ by comparison with the security of l encryptions with $\mathsf{Enc1}^{\mathsf{l}}$ performed with independent keys, block by block.

Lemma 6.6. *For every pair of l-block messages m^0 and m^1 and (s,t)-bounded adversary $\mathcal{A}^{\mathsf{L}}$, there is an $(s - s_r, t - t_r)$-bounded adversary $\mathcal{A}^{\mathsf{L}'}$ such that the following holds:*

$$\left|\Pr\left[\mathcal{A}^{\mathsf{L}}\left(\mathsf{LEncl}^{\mathsf{l}}{}_{k_0}(m^0)\right)=1\right]-\Pr\left[\mathcal{A}^{\mathsf{L}}\left(\mathsf{LEncl}^{\mathsf{l}}{}_{k_0}(m^1)\right)=1\right]\right| \leq$$
$$\sum_{i=1}^{l}\left|\Pr\left[\mathcal{A}^{\mathsf{L}'}\left(\mathsf{LEnc1}^{\mathsf{l}}{}_{k_i}(m_{0,i})\right)=1\right]-\Pr\left[\mathcal{A}^{\mathsf{L}'}\left(\mathsf{LEnc1}^{\mathsf{l}}{}_{k_i}(m_{1,i})\right)=1\right]\right|,$$

with all k's chosen uniformly at random, $s_r = l(2s_{\mathcal{S}}+1)$ and t_r equal to $2lt_{\mathcal{S}}$ to which we add the time needed to sample $2l$ random values and compute l times the $\oplus$ operations

Proof. We proceed in two steps. We start by building a sequence of $l+1$ messages $m_{h,0},\ldots,m_{h,l}$ starting from m^0 and modifying its blocks one by one until obtaining m^1. That is, $m_{h,i} := m^1_{[1,i]}, m^0_{[i+1,l]}$. From the triangle inequality, it holds that:

$$\left|\Pr\left[\mathcal{A}^{\mathsf{L}}\left(\mathsf{LEncl}^{\mathsf{l}}{}_{k_0}(m^0)\right)=1\right]-\Pr\left[\mathcal{A}^{\mathsf{L}}\left(\mathsf{LEncl}^{\mathsf{l}}{}_{k_0}(m^1)\right)=1\right]\right| \leq$$
$$\sum_{i=1}^{l}\left|\Pr\left[\mathcal{A}^{\mathsf{L}}\left(\mathsf{LEncl}^{\mathsf{l}}{}_{k_i}(m_{h,i-1})\right)=1\right]-\Pr\left[\mathcal{A}^{\mathsf{L}}\left(\mathsf{LEncl}^{\mathsf{l}}{}_{k_i}(m_{h,i})\right)=1\right]\right|$$

The l differences in the sum above can now be related to the probability of distinguishing the encryptions of single block messages: from an $\mathsf{Enc1}^{\mathsf{l}}$ encryption of $m_{0,i}$ or $m_{1,i}$ with the associated leakage $\mathsf{L}_{\mathsf{Enc1}^{\mathsf{l}}}$, it is immediate to sample an $\mathsf{Encl}^{\mathsf{l}}$ encryption of $m_{h,i-1}$ or $m_{h,i}$ with the associated leakage $\mathsf{L}_{\mathsf{Encl}^{\mathsf{l}}}$. The cost of this sampling is bounded by s_r leakage queries and running time t_r. □

Injecting Lemmas 6.4 and 6.5, which relate real and ideal encryptions, into Lemma 6.6, we obtain our main theorem for eavesdropper security.

Theorem 6.3. *Let F be a (s,t,ϵ_{prf})-PRF, with a leakage simulator $\mathcal{S}^{\mathsf{L}}$ as in Lemma 6.4, and let (s_r,t_r) be the bounds defined in Lemma 6.5. For every pair of l-block messages m^0 and m^1 and $(s-s_r,t-t_r)$-bounded adversary $\mathcal{A}^{\mathsf{L}}$, there is an $(s-2s_r,t-2t_r)$-bounded adversary $\mathcal{A}^{\mathsf{L}'}$ such that the following holds:*

$$\left|\Pr\left[\mathcal{A}^{\mathsf{L}}\left(\mathsf{LEncl}_{k_0}(m^0)\right)=1\right]-\Pr\left[\mathcal{A}^{\mathsf{L}}\left(\mathsf{LEncl}_{k_0}(m^1)\right)=1\right]\right| \leq$$
$$\sum_{i=1}^{l}\left|\Pr\left[\mathcal{A}^{\mathsf{L}'}\left(\mathsf{LEnc1}_{k_i}(m_{0,i})\right)=1\right]-\Pr\left[\mathcal{A}^{\mathsf{L}'}\left(\mathsf{LEnc1}_{k_i}(m_{1,i})\right)=1\right]\right| +$$
$$4l(\epsilon_{prf}+\epsilon_{2\text{-}sim})+negl(n),$$

where, $negl(n)$ refers to a negligible function of n assuming that l is polynomially bounded in n.

This theorem indicates that, if we want to bound the probability that an attacker distinguishes the encryptions of two l-block messages, we can focus on bounding the probabilities that an attacker distinguishes independent encryptions of the l pairs of blocks, which is arguably much easier, and derive the desired bound from this. Furthermore, the security degradation is quite moderate, being just proportional to the number of blocks.

Remark 6.6 In Theorem 6.3, we have a security degradation that is cubic in the number of message blocks. A careful analysis of the proof of LR-PRG in [SPY13, Theorem 1] suggests that this cubic degradation seems necessary when taking collisions into consideration (which seems missing in the original proof). In fact, a birthday-like attack can distinguish the intermediate hybrids H_0 and H_l in the proof of [SPY13, Theorem 1].

6.3.6 From Eavesdropper to CPA Security

The ENCI scheme is obviously insecure under chosen plaintext attack. However, the $\mathsf{PrivK}^{\mathsf{lmcpa,b}}_{\mathcal{A}^{\mathsf{L}},\mathsf{ENC}}$ security of the ENC scheme can now be derived from Theorem 6.3.

Theorem 6.4. *Let $\mathcal{A}^{\mathsf{L}}$ be an $(s - s_r, t - t_r)$-bounded $\mathsf{PrivK}^{\mathsf{lmcpa,b}}_{\mathcal{A}^{\mathsf{L}},\mathsf{ENC}}$ adversary against the ENC scheme based on a (s, t, ϵ_{prf})-secure PRF. Then,*

$$\left|\Pr[\mathsf{PrivK}^{\mathsf{lmcpa,0}}_{\mathcal{A}^{\mathsf{L}},\mathsf{ENC}} = 1] - \Pr[\mathsf{PrivK}^{\mathsf{lmcpa,1}}_{\mathcal{A}^{\mathsf{L}},\mathsf{ENC}} = 1]\right| \leq$$
$$\left|\Pr[\mathcal{A}^{\mathsf{L}\prime}(\mathsf{LEncl}_k(m^0)) = 1] - \Pr[\mathcal{A}^{\mathsf{L}\prime}(\mathsf{LEncl}_k(m^1)) = 1]\right| + 2\epsilon_{prf} + negl(n),$$

where $\mathcal{A}^{\mathsf{L}\prime}$ is $(s - 2s_r, t - 2t_r)$-bounded, m^0 and m^1 are the messages chosen by $\mathcal{A}^{\mathsf{L}}$ for the test query, $s_r := 3q$ where q is the number of encryption queries made by $\mathcal{A}^{\mathsf{L}}$, and t_r is the time needed to evaluate LEncl on q messages of at most l blocks and sample $2l$ random values. And, $negl(n)$ refers to a negligible function of n assuming that q is polynomially bounded in n.

Proof. We proceed in two steps. First, we modify the $\mathsf{PrivK}^{\mathsf{lmcpa,b}}_{\mathcal{A}^{\mathsf{L}},\mathsf{ENC}}$ game by replacing the leak-free PRF with a random function. Since the cost of the reduction of this change to the PRF security is less than the (s_r, t_r) bounds, the probability that $\mathcal{A}^{\mathsf{L}}$ detects the change is bounded by ϵ_{prf}. Next, we rely on the perfectly random distribution of the ephemeral keys (k') used by Enc to emulate the leak-free PRF and (consistently) answer all encryption queries with random IV's and random ephemeral key k_0's. For the test query, we generate a random IV, but use the LEncl oracle to produce the ciphertext. This strategy will only fail when the random IV selected here is equal to one of the IV's generated during one of the at most q previous encryption queries, bounding the probability of this event by $q/2^n$. Again, the cost of answering these queries is bounded by (s_r, t_r).

□

6.4 Conclusion and Future Directions

In this chapter, we constructed leakage-resilient MAC that is strongly unforgeable in the simulatable leakage assumption. We also constructed an LR-PRG based leakage-resilient encryption scheme also in the simulatable leakage assumption. The combination of the results in this chapter is in fact well in line with the early investigations of Micali and Reyzin, where it was shown that unpredictability-based security is easier to obtain than indistinguishability-based security in the presence of leakage [MR04].

Concretely, and based on present knowledge, it also means that if semantic (or equivalent) security is required for an application, the best option is to use leakage-resilient authentication to access a leak-free environment first, and to perform encryption only afterwards. In other words, the security guarantees of leakage-resilient encryption, despite practically meaningful (e.g. in order to prevent key recoveries), are indeed much harder to formalize in terms of message confidentiality.

Note finally that our constructions only consider the leakage-resilience of tag generation and encryption. This is a relevant first step, since it is a frequent scenario that only one (cost-constrained) party in authentication and encryption has to be protected against side-channel analysis [MSGR10]. Yet, we also admit that securing the tag verification and decryption parts will most likely be more challenging since these algorithms are not randomized in most existing MAC and encryption schemes.

A future direction would be to see if the simulatable leakage assumption could be used to construct authenticated encryption schemes.

Bibliography

[ABF13] Michel Abdalla, Sonia Belaïd, and Pierre-Alain Fouque. Leakage-resilient symmetric encryption via re-keying. In Bertoni and Coron [BC13], pages 471–488. 119

[ADW09] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In Halevi [Hal09], pages 36–54. 63, 64

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer, 2009. 8, 15

[AM11] Divesh Aggarwal and Ueli Maurer. The leakage-resilience limit of a computational problem is equal to its unpredictability entropy. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 686–701. Springer, 2011. 40, 61, 74

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2004. 6, 9, 49, 53

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *J. Cryptology*, 21(2):149–177, 2008. 35, 53, 67

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical Identity Based Encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2005. 15, 31, 33

[BC13] Guido Bertoni and Jean-Sébastien Coron, editors. *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-*

23, 2013. Proceedings, volume 8086 of *Lecture Notes in Computer Science*. Springer, 2013. 141, 153

[BDL01] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of eliminating errors in cryptographic computations. *J. Cryptology*, 14(2):101–119, 2001. 3

[BGS15] Sonia Belaïd, Vincent Grosso, and François-Xavier Standaert. Masking and leakage-resilient primitives: One, the other(s) or both? *Cryptography and Communications*, 7(1):163–184, 2015. 117, 131

[BHHG01] Dan Boneh, Shai Halevi, and Nick Howgrave-Graham. The modular inversion hidden number problem. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 36–51. Springer, 2001. 16, 17, 21

[BKKV10] Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *FOCS* [DBL10], pages 501–510. 50

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007. 106

[BKMN09] Julien Brouchier, Tom Kean, Carol Marsh, and David Naccache. Temperature attacks. *IEEE Security & Privacy*, 7(2):79–82, 2009. 3

[BN05] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford E. Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer, 2005. 42, 43, 44

[Bra39] Alfred Brauer. On addition chains. *Bull. Amer. Math. Soc*, 45(10):736–739, 1939. 87

[Bra90] Gilles Brassard, editor. *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*. Springer, 1990. 144, 153

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-*

21, 1997, Proceedings, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997. 3

[BSS05] Ian F. Blake, Gadiel Seroussi, and Nigel P. Smart. *Advances in Elliptic Curve Cryptography*, volume 317 of *London Mathematical Society Lecture Notes Series*. Cambridge University Press, 2005. 45

[BSW11] Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. In Kenneth G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 89–108. Springer, 2011. 50

[BV96] Dan Boneh and Ramarathnam Venkatesan. Hardness of computing the most significant bits of secret keys in diffie-hellman and related schemes. In Koblitz [Kob96], pages 129–142. 16, 17

[CGP+12] Claude Carlet, Louis Goubin, Emmanuel Prouff, Michaël Quisquater, and Matthieu Rivain. Higher-order masking schemes for S-Boxes. In Anne Canteaut, editor, *FSE*, volume 7549 of *Lecture Notes in Computer Science*, pages 366–384. Springer, 2012. 8, 9, 79, 81, 82, 83, 84, 85, 86, 88, 92, 94, 97, 98, 103, 104, 109

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Wiener [Wie99], pages 398–412. 8, 82, 116

[Cop97] Don Coppersmith. Small solutions to polynomial equations, and low exponent rsa vulnerabilities. *J. Cryptology*, 10(4):233–260, 1997. 17, 47

[Cor13] Jean-Sébastien Coron. `https://github.com/coron/htable/`, 2013. 111

[Cor14] Jean-Sébastien Coron. Higher order masking of look-up tables. In Nguyen and Oswald [NO14], pages 441–458. 8, 81, 111

[CPRR13] Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Higher-order side channel security and mask refreshing. In Shiho Moriai, editor, *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, volume 8424 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 2013. 81

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002. 5

[CRV14] Jean-Sébastien Coron, Arnab Roy, and Srinivas Vivek. Fast evaluation of polynomials over binary finite fields and application to side-channel countermeasures. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 170–187. Springer, 2014. Full version available at `http://eprint.iacr.org/2014/890`. 10, 95, 97

[DBL10] *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*. IEEE Computer Society, 2010. 142, 144

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Nguyen and Oswald [NO14], pages 423–440. 8

[des93] FIPS 46-3: Data Encryption Standard. National Institute of Standards and Technology, March 1993. Available via csrc.nist.gov. 108

[DF89] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Brassard [Bra90], pages 307–315. 8

[DGK+10] Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In Micciancio [Mic10], pages 361–381. 15

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 2

[DHLAW10a] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *FOCS* [DBL10], pages 511–520. 15

[DHLAW10b] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 613–631. Springer, 2010. 50

[DKXY02] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Key-insulated public key cryptosystems. In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*, pages 65–82. Springer, 2002. 8

[Dod00] Yevgeniy Dodis. *Exposure-Resilient Cryptography*. PhD thesis, Massachusetts Institute of Technology, 2000. 8

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008. 30

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302. IEEE Computer Society, 2008. 7, 15, 116, 124

[DP10] Yevgeniy Dodis and Krzysztof Pietrzak. Leakage-resilient pseudorandom functions and side-channel attacks on feistel networks. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2010. 116

[DvOW92] Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Des. Codes Cryptography*, 2(2):107–125, 1992. 8

[Fau10] Sebastian Faust. *Provable security at implementation-level*. PhD thesis, PhD thesis, Katholieke Universiteit Leuven, 2010. 6

[FH08] Julie Ferrigno and Martin Hlavác. When AES blinks: introducing optical side channel. *IET Information Security*, 2(3):94–98, 2008. 3

[FKPR10] Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, and Guy N. Rothblum. Leakage-resilient signatures. In Micciancio [Mic10], pages 343–360. 7, 50, 63, 64, 75

[FPS12] Sebastian Faust, Krzysztof Pietrzak, and Joachim Schipper. Practical leakage-resilient symmetric cryptography. In Emmanuel Prouff and Patrick Schaumont, editors, *CHES*, volume 7428 of *Lecture Notes in Computer Science*, pages 213–232. Springer, 2012. 116

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986. 6

[FT12] Pierre-Alain Fouque and Mehdi Tibouchi. Indifferentiable hashing to barreto-naehrig curves. In Alejandro Hevia and Gregory Neven, editors, *Progress in Cryptology - LATINCRYPT*

2012 - 2nd International Conference on Cryptology and Information Security in Latin America, Santiago, Chile, October 7-10, 2012. Proceedings, volume 7533 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2012. 13, 29, 42, 43, 46

[Gam85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985. 15

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178. ACM, 2009. 112

[GGL+14] David Galindo, Johann Großschädl, Zhe Liu, Praveen K. Vadnala, and Srinivas Vivek. Implementation and evaluation of a leakage-resilient ElGamal key encapsulation mechanism. In *Proceedings of PROOFS: Security Proofs for Embedded Systems, September 27, 2014, Busan, South Korea*, 2014. Full version available at `http://eprint.iacr.org/2014/835`. 9

[GHS12] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Safavi-Naini and Canetti [SNC12], pages 850–867. 112

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984. 2, 5

[GMO+14] Jake Longo Galea, Daniel P. Martin, Elisabeth Oswald, Daniel Page, Martijn Stam, and Michael Tunstall. Simulatable leakage: Analysis, pitfalls, and new constructions. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 223–242. Springer, 2014. 118

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988. 64

[Gol87] Oded Goldreich. Towards a theory of software protection and simulation by oblivious rams. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 182–194. ACM, 1987. 8

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001. 6

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004. 6

[GOPT10] Johann Großschädl, Elisabeth Oswald, Dan Page, and Michael Tunstall. Side-channel analysis of cryptographic software via early-terminating multiplications. In Donghoon Lee and Seokhie Hong, editors, *Information Security and Cryptology — ICISC 2009*, volume 5984 of *Lecture Notes in Computer Science*, pages 176–192. Springer Verlag, 2010. 42

[GPS14] Vincent Grosso, Emmanuel Prouff, and François-Xavier Standaert. Efficient masked s-boxes processing - A step forward -. In David Pointcheval and Damien Vergnaud, editors, *Progress in Cryptology - AFRICACRYPT 2014 - 7th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 28-30, 2014. Proceedings*, volume 8469 of *Lecture Notes in Computer Science*, pages 251–266. Springer, 2014. 112

[GSF14] Vincent Grosso, François-Xavier Standaert, and Sebastian Faust. Masking vs. multiparty computation: how large is the gap for AES? *J. Cryptographic Engineering*, 4(1):47–57, 2014. 116

[GST13] Daniel Genkin, Adi Shamir, and Eran Tromer. RSA key extraction via low-bandwidth acoustic cryptanalysis. *IACR Cryptology ePrint Archive*, 2013:857, 2013. 3

[GV12] David Galindo and Srinivas Vivek. A practical leakage-resilient signature scheme in the generic group model. In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, volume 7707 of *Lecture Notes in Computer Science*, pages 50–65. Springer, 2012. 9, 30, 67

[GV13] David Galindo and Srinivas Vivek. A leakage-resilient pairing-based variant of the Schnorr signature scheme. In Martijn Stam, editor, *IMA Int. Conf.*, volume 8308 of *Lecture Notes in Computer Science*, pages 173–192. Springer, 2013. 9

[GV14] David Galindo and Srinivas Vivek. Limits of a conjecture on a leakage-resilient cryptosystem. *Inf. Process. Lett.*, 114(4):192–196, 2014. 9

[Hal09] Shai Halevi, editor. *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, volume 5677 of *Lecture Notes in Computer Science*. Springer, 2009. 141, 151

[HGNS03] Nick Howgrave-Graham, Phong Q. Nguyen, and Igor Shparlinski. Hidden number problem with hidden multipliers, timed-

release crypto, and noisy exponentiation. *Math. Comput.*, 72(243):1473–1485, 2003. 17

[HGS01] Nick Howgrave-Graham and Nigel P. Smart. Lattice attacks on digital signature schemes. *Des. Codes Cryptography*, 23(3):283–290, 2001. 17

[HLWW13] Carmit Hazay, Adriana López-Alt, Hoeteck Wee, and Daniel Wichs. Leakage-resilient cryptography from minimal assumptions. In Johansson and Nguyen [JN13], pages 160–176. 119

[Ica09] Thomas Icart. How to hash into elliptic curves. In *CRYPTO*, pages 303–316, 2009. 42

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003. 7, 8, 9, 15, 81, 111, 116

[JN13] Thomas Johansson and Phong Q. Nguyen, editors. *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*. Springer, 2013. 148, 151

[Kah96] David Kahn. *The Codebreakers: The comprehensive history of secret communication from ancient times to the internet.* Simon and Schuster, 1996. 2

[Kat09] Jonathan Katz. Signature schemes with bounded leakage resilience. Cryptology ePrint Archive, Report 2009/220, 2009. `http://eprint.iacr.org/`. 63, 64

[Kiz11] Ilya Kizhvatov. *Physical Security of Cryptographic Algorithm Implementations.* PhD thesis, University of Luxembourg, June 2011. 3, 4, 6

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Wiener [Wie99], pages 388–397. 3, 4

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography.* Chapman and Hall/CRC Press, 2007. 6

[KM07] Neal Koblitz and Alfred Menezes. Another look at generic groups. *Adv. in Math. of Comm.*, 1(1):13–28, 2007. 6

[Knu97] Donald E. Knuth. *The Art of Computer Programming, Volume II: Seminumerical Algorithms, 3rd Edition.* Addison-Wesley, 1997. 82, 84, 86, 87, 91

[Kob96] Neal Koblitz, editor. *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, vol-

ume 1109 of *Lecture Notes in Computer Science*. Springer, 1996. 143, 149

[Koc96] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Koblitz [Kob96], pages 104–113. 3

[KP10a] Eike Kiltz and Krzysztof Pietrzak. Leakage resilient ElGamal encryption. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 595–612. Springer, 2010. 7, 9, 13, 15, 16, 18, 19, 20, 28, 29, 31, 33, 34, 41, 45, 47, 48, 49, 50, 51, 57, 72, 117, 149

[KP10b] Eike Kiltz and Krzysztof Pietrzak. Leakage resilient ElGamal encryption, 2010. Slides of ASIACRYPT 2010 presentation. 20

[KP10c] Eike Kiltz and Krzysztof Pietrzak. Leakage resilient ElGamal encryption. Full version of [KP10a]. `http://homepage.ruhr-uni-bochum.de/Eike.Kiltz/papers/elgamal_leak.pdf`. Last accessed June 04, 2014, 2010. 29, 34

[KR11] Lars R. Knudsen and Matthew Robshaw. *The Block Cipher Companion*. Information Security and Cryptography. Springer, 2011. 116

[KV09] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 703–720. Springer, 2009. 50

[LL12] Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In Safavi-Naini and Canetti [SNC12], pages 517–532. 7, 18

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982. 23

[LSSW12] San Ling, Igor Shparlinski, Ron Steinfeld, and Huaxiong Wang. On the modular inversion hidden number problem. *J. Symb. Comput.*, 47(4):358–367, 2012. 48

[Man02] Stefan Mangard. A simple power-analysis (SPA) attack on implementations of the AES key expansion. In Pil Joong Lee and Chae Hoon Lim, editors, *Information Security and Cryptology - ICISC 2002, 5th International Conference Seoul, Korea, November 28-29, 2002, Revised Papers*, volume 2587 of *Lecture Notes in Computer Science*, pages 343–358. Springer, 2002. 4

[Mau05] Ueli M. Maurer. Abstract models of computation in cryptography. In Nigel P. Smart, editor, *IMA Int. Conf.*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2005. 6, 35, 38, 53, 56, 67, 70

[Mes00] Thomas S. Messerges. Using second-order power analysis to attack DPA resistant software. In Çetin Kaya Koç and Christof Paar, editors, *CHES*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 2000. 4, 80

[Mic10] Daniele Micciancio, editor. *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, volume 5978 of *Lecture Notes in Computer Science*. Springer, 2010. 144, 145

[Mon85] Peter L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170):519–521, April 1985. 46

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer Verlag, 2007. 4, 6

[Mor12] Amir Moradi. Statistical tools flavor side-channel collision attacks. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 428–445. Springer, 2012. 28

[MOS13] Daniel P. Martin, Elisabeth Oswald, and Martijn Stam. A leakage resilient MAC. *IACR Cryptology ePrint Archive*, 2013:292, 2013. 117, 118, 120

[MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2004. 7, 15, 18, 34, 139

[MSGR10] Marcel Medwed, François-Xavier Standaert, Johann Großschädl, and Francesco Regazzoni. Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In Daniel J. Bernstein and Tanja Lange, editors, *Progress in Cryptology - AFRICACRYPT 2010, Third International Conference on Cryptology in Africa, Stellenbosch, South Africa, May 3-6, 2010. Proceedings*, volume 6055 of *Lecture Notes in Computer Science*, pages 279–296. Springer, 2010. 139

[MTVY11] Tal Malkin, Isamu Teranishi, Yevgeniy Vahlis, and Moti Yung. Signatures resilient to continual leakage on memory and computation. In Yuval Ishai, editor, *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 89–106. Springer, 2011. 50

[MvV96] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. 1, 2

[NGG09] Yassir Nawaz, Kishan Chand Gupta, and Guang Gong. Algebraic immunity of s-boxes based on power mappings: analysis and construction. *IEEE Transactions on Information Theory*, 55(9):4263–4273, 2009. 82

[Ngu10] Phong Q. Nguyen. Hermite's constant and lattice algorithms. In Phong Q. Nguyen and Brigitte Vallé, editors, *The LLL Algorithm*, Information Security and Cryptography, pages 19–69. Springer Berlin Heidelberg, 2010. 23

[NO14] Phong Q. Nguyen and Elisabeth Oswald, editors. *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*. Springer, 2014. 143, 144

[NRS11] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure hardware implementation of nonlinear functions in the presence of glitches. *J. Cryptology*, 24(2):292–321, 2011. 116

[NS02] Phong Q. Nguyen and Igor Shparlinski. The insecurity of the digital signature algorithm with partially known nonces. *J. Cryptology*, 15(3):151–176, 2002. 17

[NS09] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In Halevi [Hal09], pages 18–35. 8, 15, 119

[NSW09] Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Hash function requirements for schnorr signatures. *J. Mathematical Cryptology*, 3(1):69–87, 2009. 64, 65, 66, 67

[Oka92] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53. Springer, 1992. 64

[Pie09] Krzysztof Pietrzak. A leakage-resilient mode of operation. In Antoine Joux, editor, *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 462–482. Springer, 2009. 7, 116

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Johansson and Nguyen [JN13], pages 142–159. 8

[PS73] Mike Paterson and Larry J. Stockmeyer. On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM J. Comput.*, 2(1):60–66, 1973. 95, 99

[PSMD14] Peter Pessl, François-Xavier Standaert, Stefan Mangard, and François Durvaux. Towards leakage simulators that withstand the correlation distinguisher. *ASIACRYPT 2014 rump session talk*, 2014. 118

[PSNB11] Geovandro C. Pereira, Marcos A. Simplício, Michael Naehrig, and Paulo S. Barreto. A family of implementation-friendly BN elliptic curves. *Journal of Systems and Software*, 84(8):1319–1326, August 2011. 44

[PSP+08] Christophe Petit, François-Xavier Standaert, Olivier Pereira, Tal Malkin, and Moti Yung. A block cipher based pseudo random number generator secure against side-channel key recovery. In Masayuki Abe and Virgil D. Gligor, editors, *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2008, Tokyo, Japan, March 18-20, 2008*, pages 56–65. ACM, 2008. 116

[PSV15] Olivier Pereira, François-Xavier Standaert, and Srinivas Vivek. Leakage-resilient authentication and encryption from symmetric primitives, 2015. Manuscript under submission. 10

[PV04] Dan Page and Frederik Vercauteren. Fault and side-channel attacks on pairing based cryptography. *IACR Cryptology ePrint Archive*, 2004:283, 2004. 47

[QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In Isabelle Attali and Thomas P. Jensen, editors, *Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001. 3

[RDP08] Matthieu Rivain, Emmanuelle Dottax, and Emmanuel Prouff. Block ciphers implementations provably secure against second order side channel analysis. In Kaisa Nyberg, editor, *FSE*, volume 5086 of *Lecture Notes in Computer Science*, pages 127–143. Springer, 2008. 82

[Roy13] Arnab Roy. *Security Aspects of Symmetric-Key Primitives*. PhD thesis, University of Luxembourg, April 2013. 10

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of aes. In Stefan Mangard and François-Xavier Standaert, editors, *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010. 8, 81

[RV13] Arnab Roy and Srinivas Vivek. Analysis and improvement of the generic higher-order masking scheme of FSE 2012. In

Bertoni and Coron [BC13], pages 417–434. Full version available at `http://eprint.iacr.org/2013/345`. 10, 87, 95, 106, 108

[Sch80] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980. 30

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Brassard [Bra90], pages 239–252. 64

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991. 63, 64

[Sch10] Joachim Schipper. Leakage-resilient authentication. *M.Sc. thesis, Centrum Wiskunde and Informatica, The Netherlands*, 2010. 118, 122

[Sco05] Michael Scott. Computing the tate pairing. In Alfred Menezes, editor, *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 293–304. Springer, 2005. 41, 46

[Sco11] Michael Scott. On the efficient implementation of pairing-based protocols. In Liqun Chen, editor, *IMA Int. Conf.*, volume 7089 of *Lecture Notes in Computer Science*, pages 296–308. Springer, 2011. 75

[Sha49] Claude E Shannon. Communication theory of secrecy systems*. *Bell system technical journal*, 28(4):656–715, 1949. 5

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997. 6, 15, 18, 31, 33, 35, 38, 53, 56, 64, 67, 70

[Shp05] Igor E Shparlinski. Playing "hide-and-seek" with numbers: the hidden number problem, lattices, and exponential sums. In *Symposia in Applied Mathematics*, volume 62, pages 153–177. American Mathematical Society, 2005. 16, 17

[SNC12] Reihaneh Safavi-Naini and Ran Canetti, editors. *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*. Springer, 2012. 146, 149

[SP06] Kai Schramm and Christof Paar. Higher order masking of the aes. In David Pointcheval, editor, *CT-RSA*, volume 3860 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2006. 82

[SPY+10] François-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. Leakage resilient cryptography in practice. In Ahmad-Reza Sadeghi

and David Naccache, editors, *Towards Hardware-Intrinsic Security - Foundations and Practice*, Information Security and Cryptography, pages 99–134. Springer, 2010. 116

[SPY13] François-Xavier Standaert, Olivier Pereira, and Yu Yu. Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 335–352. Springer, 2013. 8, 10, 15, 28, 116, 118, 123, 125, 131, 138

[SSA+07] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. The 128-bit blockcipher CLEFIA (extended abstract). In Alex Biryukov, editor, *FSE*, volume 4593 of *Lecture Notes in Computer Science*, pages 181–195. Springer, 2007. 105

[ST06] Douglas Stebila and Nicolas Thériault. Unified point addition formulæ and side-channel attacks. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems — CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 354–368. Springer Verlag, 2006. 42

[Sti95] Douglas R. Stinson. *Cryptography - theory and practice*. Discrete mathematics and its applications series. CRC Press, 1995. 2

[SvdW06] Andrew Shallue and Christiaan van de Woestijne. Construction of rational points on elliptic curves over finite fields. In Florian Hess, Sebastian Pauli, and Michael E. Pohst, editors, *ANTS*, volume 4076 of *Lecture Notes in Computer Science*, pages 510–524. Springer, 2006. 42

[The12] The PARI Group, Bordeaux. *PARI/GP, version* `2.3.4`, 2012. `http://pari.math.u-bordeaux.fr/`. 25

[vzG91] Joachim von zur Gathen. Efficient and optimal exponentiation in finite fields. *Computational Complexity*, 1:360–394, 1991. 84, 85, 86, 87

[vzGN97] Joachim von zur Gathen and Michael Nöcker. Exponentiation in finite fields: Theory and practice. In Teo Mora and Harold F. Mattson, editors, *AAECC*, volume 1255 of *Lecture Notes in Computer Science*, pages 88–113. Springer, 1997. 82

[vzGN04] Joachim von zur Gathen and Michael Nöcker. Computing special powers in finite fields. *Math. Comput.*, 73(247):1499–1523, 2004. 87

[Wal04] Colin D. Walter. Simple power analysis of unified code for ECC double and add. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems —*

CHES 2004, volume 3156 of *Lecture Notes in Computer Science*, pages 191–204. Springer Verlag, 2004. 42, 47

[Wic11] Daniel Wichs. *Cryptographic Resilience to Continual Information Leakage*. PhD thesis, New York University, 2011. 8

[Wic13] Daniel Wichs. Barriers in cryptography with weak, correlated and leaky sources. In Robert D. Kleinberg, editor, *ITCS*, pages 111–126. ACM, 2013. 9, 15, 63

[Wie99] Michael J. Wiener, editor. *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*. Springer, 1999. 143, 148

[WS06] Claire Whelan and Michael Scott. Side channel analysis of practical pairing implementations: Which path is more secure? In Phong Q. Nguyen, editor, *VIETCRYPT*, volume 4341 of *Lecture Notes in Computer Science*, pages 99–114. Springer, 2006. 47

[WT01] Colin D. Walter and Susan Thompson. Distinguishing exponent digits by observing modular subtractions. In David Naccache, editor, *Topics in Cryptology — CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 192–207. Springer Verlag, 2001. 42

[YS13] Yu Yu and François-Xavier Standaert. Practical leakage-resilient pseudorandom objects with minimum public randomness. In Ed Dawson, editor, *Topics in Cryptology - CT-RSA 2013 - The Cryptographers' Track at the RSA Conference 2013, San Francisco,CA, USA, February 25-March 1, 2013. Proceedings*, volume 7779 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2013. 116

[YSPY10] Yu Yu, François-Xavier Standaert, Olivier Pereira, and Moti Yung. Practical leakage-resilient pseudorandom generators. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 141–151. ACM, 2010. 116

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In Edward W. Ng, editor, *EUROSAM*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979. 30

Publications

Journal Publications

1. *with* Jean-Sébastien Coron and Arnab Roy. **Fast Evaluation of Polynomials over Binary Finite Fields and Application to Side-channel Countermeasures**. *J. Cryptographic Engineering*, Springer. Volume 5, Issue 2 (2015) 73-83.
2. *with* C. E. Veni Madhavan. **Cubic Sieve Congruence of the Discrete Logarithm Problem, and fractional part sequences**. *J. Symbolic Computation*, Elsevier. Volume 64 (August 2014) 22-34.
3. *with* David Galindo. **Limits of a Conjecture on a Leakage-Resilient Cryptosystem**. *Information Processing Letters*, Elsevier. Volume 114, Issue 4 (2014) 192-196.

Peer-Reviewed Conference Proceedings

4. *with* Olivier Pereira and François-Xavier Standaert. **Leakage-Resilient Authentication and Encryption from Symmetric Primitives**. *Manuscript under submission.*
5. *with* David Galindo, Johann Großschädl, Zhe Liu, and Praveen Kumar Vadnala. **Implementation and Evaluation of a Leakage-Resilient ElGamal Key Encapsulation Mechanism**. In proc. of *PROOFS 2014*. Available at `http://eprint.iacr.org/2014/835`. **(Invited for submission to J. Cryptographic Engineering.)**
6. *with* Jean-Sébastien Coron and Arnab Roy. **Fast Evaluation of Polynomials over Binary Finite Fields and Application to Side-channel Countermeasures**. In proc. of *CHES 2014*. *LNCS* 8731 (2014) 170-187. **(Rated among the top 3 papers.)**
7. *with* David Galindo. **A Leakage-Resilient Pairing-Based Variant of the Schnorr Signature Scheme**. In proc. of *IMA CC 2013*. *LNCS* 8308 (2013) 173-192.
8. *with* Arnab Roy. **Analysis and Improvement of the Generic Higher-Order Masking Scheme of FSE 2012**. In proc. of *CHES 2013*. *LNCS* 8086 (2013) 417-434.

9. *with* David Galindo. **A Practical Leakage-Resilient Signature Scheme in the Generic Group Model**. In proc. of *SAC 2012*. *LNCS* 7707 (2013) 50-65.

Unrefereed Publication

10. *with* Shankar B. R. **Integer Complexity: Breaking the $\Theta(n^2)$ barrier**. *Eprint*. Available at `http://hdl.handle.net/10993/18508`.